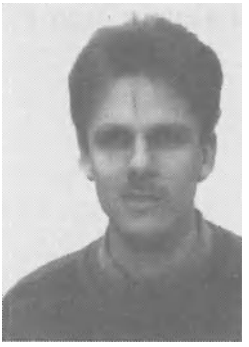
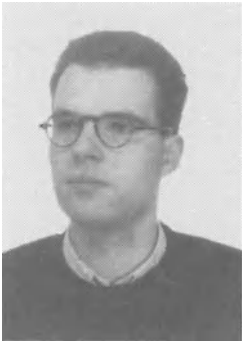
LINUX - Unleashing the Workstation in Your PC





The authors,
Stefan Strobel and
Thomas Uhl,
are senior students of
medical informatics at
the University of
Heidelberg and
Heilbronn Polytechnic
in Germany.
They have been
administrating the
workstations and
Linux PC's installed
at Heilbronn College
for some time.
In teaching numerous
college courses, they have
introduced a wide
audience to Linux and
thus promoted its
propagation.

Stefan Strobel Thomas Uhl

LINUX
Unleashing the Workstation in Your PC

Foreword by Jürgen Gulbins
Translation by Robert Bach

Springer-Verlag
Berlin Heidelberg New York
London Paris Tokyo
Hong Kong Barcelona
Budapest

Stefan Strobel
Schlegelstraße 19
D-74074 Heilbronn
Germany

Thomas Uhl
Obere Heerbergstraße 17
D-97078 Würzburg
Germany

The authors can be reached at the following e-mail addresses:

linux@fh-heilbronn.de
stefan.strobel@linux.org
thomas.uhl@linux.org

ISBN-13: 978-3-540-58077-5 e-ISBN-13: 978-3-642-97572-1
DOI:10.1007/978-3-642-97572-1

CIP data applied for

Cover Design: Konzept & Design, Ilvesheim
Typesetting: Camera-ready by authors
SPIN 10471546 33/3140 – 5 4 3 2 1 – Printed on acid-free paper

Foreword

UNIX achieved its widespread propagation, its penetration of the university domain, and its reach into research and industry due to its early dissemination by AT&T to all interested parties at almost no cost and as source code. UNIX's present functionality emanated not just from AT&T developers but also from external developers who used the product and contributed their own further developments, which they then put at AT&T's disposal. (Consider the contributions of the University of California at Berkeley, for example.) With the rising commercialization of UNIX by AT&T (now by Novell) since 1983, such creative and cooperative further development became increasingly restricted, and UNIX source code today has become unaffordably expensive and scarcely accessible.

UNIX history

Linux provides interested computer scientists and users with a system that revives the old UNIX tradition: Linux is available for free, and everyone is heartily invited (but not obliged) to contribute to its further development. Since Linux runs on PC systems, it has begun to penetrate the workrooms of many computer science students and computer freaks.

free & participatory

The functionality and completeness that Linux has achieved by far outstrips a mere toy or demonstration system. In many aspects, even if perhaps not all, Linux can hold its own against proprietary UNIX versions that are seen as realistic, such as those from Sun, HP, IBM, and SCO. The successful porting of a large number of small and medium-sized applications as well as the availability of the current version of the X Window System under Linux illustrate this point.

free software problems

However, due to the lack of financing by means of contributions, subsidies or sales, free software like Linux faces many problems. For example, free software often lacks mature, full-fledged documentation. Therefore this book strives to complement the readily available Linux Documentation Pages by means of an overview and numerous notes and additions. This will permit the Linux novice a rapid initiation, better utilization of the system, and a look at areas that were unfamiliar or could not be found. Especially the instructions for installation and administration—which unfortunately vary greatly from one UNIX system to the next—will prove helpful to most Linux users.

pioneering spirit

I sincerely hope that Linux rekindles the pioneering spirit of the early UNIX years and really shows what is possible and useful, for the benefit of the commercial UNIX peddlers with their often ridiculous, unresolved, and truly proprietary discussions. May this book contribute to this effort.

J. Gulbins

Note from the Authors

This book has evolved from the experience that we gained through several courses that we taught and from the questions that we repeatedly encounter on the subject of Linux. We have seen that novices do not necessarily need a multitude of technical details to get started, but a broad overview of the Linux system and the many freely available tools. As such this book does not serve as a reference, but imparts to the reader the necessary fundamental knowledge and makes him/her able to seek further sources of information.

Acknowledgments

We wish to expressly convey our gratitude to the following persons, who actively contributed to the production of this book: Dirk Höfle, Sascha Runge, Peter Welker and Roland Uhl. We owe special thanks to the director of the computing center at Heilbronn Polytechnic, Dr. G. Peter, and his staff. We are obliged to J. Gulbins for contributing the foreword.

We also thank our translator, Bob Bach, for the synergy and the brutal night shifts that he shared with us in the final stages of preparing the manuscript for publication.

This book could never have been written without the preliminary work of Linus Torvalds and the international Linux network community that combined its energy to make Linux what it is today, and the pioneering efforts of Richard Stallman, founder of the Free Software Foundation, and the dedicated FSF network community that gave Linux a home in a broad base of software support.

Contents

Introduction

The Basics

Networking

Linux Features

Installation

Configuration

Administration

Support and Help

X Window System

Languages & Tools

Applications

Network Applications

Appendix

Index

Introduction

For some time 32-bit machines have been a hot topic in the world of PCs. It seems that more powerful operating systems will soon be displacing DOS. At least in the professional literature, lively discussion is raging about which operating system will assert itself as the future standard. Two alternatives seem to be emerging for the domain of server operating systems: Windows NT, and UNIX variants such as Solaris 2, UnixWare (Novell), and NextStep. In this context OS/2 plays no significant role since it is seen more as a competitor to Windows in its current version and future 32-bit versions.

UNIX vs. Windows NT

We cannot yet predict which system will finally predominate. However, the significant rise in the power of hardware in recent years has unleashed the demand for a modern operating system that makes use of these developments. Under a modern server operating system, the borderline between classical UNIX workstations and high-end PCs will tend to become more fluid.

1.1 Historical perspectives

An extremely powerful alternative to the above proprietary systems has evolved far from all the big debates on strategy. The system is Linux, a UNIX system for Intel processors that is available for free.

Linux was developed by a young Finnish student named Linus Torvalds. His initial goal was not to develop a full-scale operating system. He only wanted to acquaint himself with the special task-switching commands of the 80386 processor. To

386-Processor

compile his test program, he used MINIX, a pedagogical operating system by Andrew Tanenbaum.

Yet, due to its didactic orientation, MINIX had some shortcomings. The ambitious student soon exhausted the possibilities of the UNIX-like system. From his test program, he began to develop stepwise a small kernel that ran in the protected mode of the 80386 and thus optimally exploited the processor.

keyboard driver

After the task switcher, Linus Torvalds wrote a simple keyboard driver to allow him to work interactively with the system. At this point Linux still relied on parts of the MINIX system, but that was soon to change.

minix file system

To avoid having to develop a new file system as well, Linus Torvalds decided to adopt the MINIX file system. This not only saved him a great deal of work, but also provided from the start a stable system for managing the hard disk. After a few months the developer considered the system to be mature enough to present it to a more general public.

first version

In August 1991 the complete source code of Linux appeared for the first time on Finland's largest ftp server (Internet address `nic.funet.fi`). It was announced as a "freely distributable MINIX clone" and caught the attention of only a few interested parties on the net. In October the next version (0.02) was published, containing some rudimentary UNIX commands. The accompanying GNU compiler (`gcc`) permitted the compilation of small C programs and thus enabled the porting of a UNIX shell (`bash`).

POSIX

The early decision to adhere to POSIX, a family of standards of the Institute of Electrical and Electronics Engineers (IEEE), played a deciding role in assuring the portability of standard UNIX software to today's Linux. However, it took until the end of the year before Linux received more widespread notice. The breakthrough came January 5, 1992, with version 0.12. Linux had attained sufficient power to interest a larger community of developers. The system had also acquired a swap mechanism that gave it an unequivocal edge over MINIX.

Over time an ever-increasing number of interested developers began sending corrections and suggestions to Finland, thus participating in the improvement of the Linux system. Early

developments contributed in this way include the POSIX Job Control in version 0.12 and the switchable virtual consoles.

pronunciation

Note: Even insiders often mispronounce the word Linux. Many users consider it an American term and pronounce it with an English long i and short u. The Finnish pronunciation is the correct one, amounting to "lee-nooks" for an English speaker.

The Internet proved to be an important tool for the rapid development of Linux. The Internet is the superset of many wide-area networks (WANs) and "information highways" connecting more than a million computers around the world and allowing the fast exchange of information of all kinds. The Internet permits Linux developers to exchange comments, improvements and programs.

Internet

Early on Linus Torvalds was bombarded with over sixty electronic mail messages per day that he could hardly read and answer. Only after several discussion groups for Linux were set up did the flood of mail subside. Today there are several newsgroups concerned with Linux. The most important is comp.os.linux.announce (c.o.l.a.), where new developments and program versions are announced (see Section 8.3).

newsgroups

Mailing lists were set up for Linux developers to permit a similar kind of information exchange. To enable better selection according to specialized domains, several channels were established, each of which discusses a particular subject (kernel, X11, compilers, etc.).

mailing lists

In addition to letters and information, files can be exchanged via the Internet. This makes it possible to organize distributed development of larger software systems, as Linux so impressively demonstrates.

The rapid flow of information proves to be a tangible advantage not only for developers but also for users of Linux. If the user encounters any problems during installation or detects errors during operation, then with a little luck and the Internet, an adequate solution is only a couple of hours away. Even commercial service contracts seldom provide such extensive support.

support

Naturally not every Linux user has access to the Internet, but even then the user is not deserted. Many BBS (Bulletin Board System) networks have set up Linux discussion groups, so that a modem suffices for keeping on top of things.

no authority

An interesting aspect of Linux history is that there was never a strict authority hierarchy that managed the development in any way. Rather, the project has been fueled by the enthusiasm of many individual Internetters who continue to contribute new improvements and suggestions. These are often professional developers or employees of large institutions who contribute their free time.

GNU C and
X Window System

Although Linus Torvalds continues to handle the concrete further development of the kernel, several competent allies have taken over other areas of the system. Such areas include the porting and maintenance of the GNU C compiler and the C libraries for Linux, the maintenance and adaptation of the X Window System, and networking. Other Linux devotees are working on user and system documentation, or assembling an installable system on diskettes or CD-ROM.

Linux Software Map

Free source code is available not only for the kernel but also for most application programs. They come primarily from the huge UNIX freeware archives on the Internet. Periodically a software catalog (Linux Software Map) is distributed via the Internet which currently contains some 1300 software packages. There is scarcely a domain for which some suitable software cannot be found.

Ingres, Postgres

Since much development at American universities is carried out with UNIX, and such developments become public domain, many implementations in the area of research are available for Linux. One example is compilers for well-known programming languages as well as obscure ones. Also, the database systems Ingres and Postgres from the University of California at Berkeley have been ported to Linux.

Although the emphasis remains on freeware, commercial applications are also available, such as a Modula-2 compiler, a Smalltalk development system, the graphical user interface

builder from ParcPlace, several database systems, and OSF/Motif user interface that has become a standard in the UNIX world.

The further development of Linux is currently experiencing big leaps similar to the first implementation of the kernel. Version 1.0 was scheduled for December 1992, but was delayed—not because of a lack of stability, but because the functionality had not yet matched that of proprietary UNIX systems.

version 1.0

The release of version 1.0 was repeatedly delayed. In retrospect, Linus Torvalds says that he should have declared the first stable and usable version 0.12 as version 1.0, for the zero in the version number apparently scared off many potentially interested parties from delving deeper into the system.

Finally released early in 1994, version 1.0 has all the important features of its proprietary competitors. Linux can boast of one important advantage over the competitors: due to its efficient design, it extracts much higher performance from a given hardware configuration. This applies to the graphical user interface as well as the kernel.

A feature that enhances the use of Linux in commercial domains is the ability to run programs in COFF or ELF format (see page 54). This emulator opens a nearly unlimited world of professional applications for the Linux user.

COFF, ELF

Contrary to many other UNIX systems, Linux already employs the newest release of the X Window System (X11R6). Additional features that proprietary UNIX systems seldom provide include the support of INMOS transputer boards and the option to run TCP/IP via the serial or parallel port. Future plans include direct kernel support for ISDN boards for fast network connections over long distances. However, since there is no road map for the further development of Linux, the system will surely provide some surprises along the way.

X Window System

1.2 UNIX history and standards

The history of UNIX goes back well into the Seventies. Dennis Ritchie and Ken Thompson at AT&T Bell Laboratories developed the first version in 1971. With the availability in 1973

Ritchie, Thompson

of the compiler language C, most of the UNIX system was rewritten, which later proved to be a great advantage for porting the system to other processors.

Due to an agreement with the U.S. government, AT&T could not market their by then quite successful system. Therefore AT&T gave UNIX as source code, although without support, to universities, where its popularity grew. With Version 7 in 1979 AT&T announced a change in their licensing policy: UNIX source code would only be provided for a fee. This prompted the University of California at Berkeley to develop their own variant, BSD (Berkeley Software Distribution) UNIX. In 1983 AT&T announced the marketing of its enhanced System V. The System V Interface Definition defined the programming interface to this system.

AT&T

BSD UNIX

System V

Companies like Sun Microsystems, Microsoft and DEC developed their own versions of UNIX (SunOS, Xenix, ULTRIX), which in time unnecessarily encumbered the porting of software between these systems. In order to merge the two main flavors of UNIX (BSD and System V), in 1990 AT&T propagated its Release 4 of System V as a new standard that encompasses all previous variants of UNIX.

Other institutions also have recognized the need for a standardization of UNIX. The Institute of Electrical and Electronic Engineers (IEEE) developed the POSIX standard for UNIX-related operating systems. This standard is divided into several parts. POSIX 1003.1 describes only the lowest-level system interface; 1003.2 defines a standard for shells and commands; 1003.7 covers the possibilities of system administration. Although POSIX actually bases on the UNIX system interface, this standard will also be supported by other operating systems (e.g. Windows NT). Linux adheres to the POSIX standard.

POSIX

Another body, consisting primarily of UNIX manufacturers, has released another standard. Although the X/Open Portability Guide bases on POSIX 1003.1, it provides extensions in certain points. Within the realm of the COSE Initiative, the importance of this organization rose significantly. The present goal is to release a uniform desktop user interface and programmer

X/Open

COSE

interface for all available UNIX variants. This should drastically reduce the effort required for porting software.

1.3 The Free Software Foundation

GNU project

Many Linux system components originated from the GNU project of the Free Software Foundation (FSF), which thereby made a valuable contribution to the development of the Linux system. FSF was founded by Richard Stallman, the developer of the legendary GNU Emacs editor. The organization "aims to make high-quality free software available to everyone." The FSF states explicitly that "free" in their title refers to "freedom", not "zero dollars".

GPL

The FSF's exact conditions are set down in their GNU General Public License (GPL), sometimes called *copyleft*. Software that has the FSF copyleft can be sold commercially, but must be delivered complete with source code. This does not apply to programs that were developed with the GNU compiler if they do not use routines from the GNU libraries.

The FSF's largest project is the implementation of a freeware operating system that is to be largely UNIX-compatible. (By the way, GNU stands for "Gnu's not UNIX.") The GNU C/C++ compiler, which is available for most hardware platforms, was developed in the realm of this project.

GNU C / C++

Numerous programmers all over the world have joined forces in the spirit of free software and are promoting the GNU project with the development of UNIX-compatible commands, along with other software. Interested programmers can discover the overwhelming multitude of software that is bound by the GNU General Public License on large ftp servers. In addition to numerous UNIX commands and programming languages like C, C++, Smalltalk, Lisp and FORTRAN, these servers provide editors, debuggers (gdb), and even a PostScript interpreter (Ghostscript).

Ghostscript

Although Linux is not a direct development of the FSF, the Foundation has been of extreme importance to Linux. Large parts of Linux also fall under the FSF's copyleft, and Linux contains

numerous commands, editors and compilers that actually evolved from the GNU project.

1.4 An overview of Linux features

To give the reader a better orientation, we offer the following summary of Linux's most important features:

- **A full-fledged 32-bit multi-user/multitasking UNIX system**. Linux permits multiple users to execute (different) programs simultaneously and thereby fully exploits the capacity of the Intel 80386 processor and its successors. The resulting performance is definitely comparable to a classical RISC workstation.
- **Orientation to common UNIX standards** (POSIX). By adhering to existing standards for UNIX, available software usually can be ported to Linux without problems.
- **Network support** (TCP/IP). A Linux machine can easily be integrated into a TCP/IP network. Linux supports common Ethernet boards and TCP/IP connection via modem (SLIP, PPP).
- **Graphical user interface** (X Window System). The Linux system includes the current version (release 6) of the X Window System. OSF/Motif, the standard user interface for proprietary UNIX systems, is also available.
- **GNU utilities and programs**. Many of Linux's commands and utilities emanate from the GNU project and contribute much functional enhancement.
- **Complete UNIX development environment**. Linux permits the development of programs that run problem-free on other UNIX systems. In addition to the GNU C/C++/Objective C compiler, numerous editors, and several version control systems, there are numerous other software development tools.

Chapter 2

The Basics

In order to understand the following chapters, the reader will need some knowledge of computer science in general and UNIX in particular. To ease the transition into this material for readers who are newcomers to UNIX, we present some of the most important concepts and terms in this chapter.

2.1 Multi-user operation

mainframe

In classical data processing (DP), a central mainframe handles all required DP tasks. Serial connections link terminals (simple text-oriented consoles with keyboards) to this mainframe. Many users sharing a single mainframe necessitates a system of access control and user management to achieve fair distribution of the shared resources.

user, user ID, password

Unique user names, also known as *user IDs* (identification), form the basis for such a system. Every user is assigned such a user ID; the user must provide the system with this user ID and its associated *password*, a procedure known as *logging in* to the system. On the basis of the user ID and its associated *access privileges*, or *permissions*, the system appropriates access for users to files and other resources.

multi-user system

Systems that provide such management for multiple users are known as multi-user systems. UNIX is a typical multi-user system, providing for each user a user ID and one or more group IDs. A user can simultaneously belong to multiple groups. This is practical if the user is involved in multiple projects or has to have

access to data in different domains. Permissions for directories and files can be assigned individually for users and groups.

All multi-user systems have one privileged user who administers the system. This user is called the system administrator or superuser. Under UNIX this privileged user has the name *root*. This user's permissions are unlimited. The administrator can create new users and assign permissions. Chapter 7 explains the tasks of the system administrator in more detail.

superuser, root

In recent years hardware prices have fallen while performance rose. The possibilities afforded by decentralized systems with their graphical user interfaces led to a decline in the acceptance of mainframes. Instead of a central mainframe with many terminals and numerous users, today's trends reveal an increasing number of workplaces where each user has sole use of one or more computers. These computers are connected via a network to each other and to other computers and servers to permit the easy exchange of data.

network

A similar development has occurred in the area of UNIX: the purely text-oriented UNIX system has evolved into a graphical desktop UNIX workstation.

The trend that a single user might use multiple computers at the same time in order to simultaneously access various programs and data is also reflected in *virtual terminals*. They allow a user to log in to the same computer repeatedly, and then switch among various virtual terminals with a special key combination.

virtual terminals

2.2 Multitasking

Newer multi-user systems are usually multitasking systems as well. They afford the possibility to process many tasks more or less simultaneously, which simple multi-user systems cannot necessarily do.

The smallest unit that can be handled in parallel in such a system is called process or task. In UNIX, which is both a multi-user and a multitasking system, we generally speak of processes. Processes that execute in parallel in a UNIX system could be

processes and daemons

programs from different users or programs that always run in the background (daemons).

A significant characteristic of modern multitasking systems is the availability of interprocess communication (IPC). This encompasses functions for synchronization or for data exchange between processes.

IPC

On computers with only one processor, the CPU must be allocated alternatingly among the individual processes in order to give the user the impression of simultaneous execution. This task is assumed by the scheduler, a special process that maintains a list of the normal processes and sees to it that the processor handles the next process at certain time intervals.

scheduler

There are various strategies that a scheduler can use to determine which process to handle next. One very simple strategy (round robin) selects the next process in the list at regular intervals (e.g., 50 ms) and puts it at the end of the list after the allocated time if the process has not yet finished. Another strategy assigns each process a priority, whereby processes with higher priority are allocated more CPU time.

UNIX employs *nice levels*, which allow the user to influence the internal priorities of processes. This allows the user to significantly reduce the encumbering of the system by programs running in the background. Likewise, the system administrator can also raise the priority of important processes to assure faster execution.

nice level

2.3 Memory management

Memory management in today's UNIX systems differs substantially from that of a simpler operating system such as DOS. Linux employs virtual memory management, which means that the operating system pretends to provide more main memory to the programs than is actually available.

The method used to implement virtual memory management in Linux is called *paging*. With the help of tables, the operating system maps a large logical address space onto a smaller physical address space. When processes demand more main memory than is physically present, individual pages of logical memory that

paging

have not been referenced recently are relocated onto the hard disk.

When a program accesses a logical address that is currently located on the hard disk, the respective memory segment (called a *page*) is loaded into main memory, while another memory segment must be written to the hard disk to compensate. Due to the significantly higher access time of a hard disk compared to main memory, there is naturally a price to be paid in terms of speed of execution.

swap file/partition

In order to be able to use the hard disk for virtual memory management and the logical main memory, swap files or swap partitions must be created on the hard disk. Without such partitions or files, main memory is limited to its actually available physical size.

2.4 Shell model

The structure of a UNIX system is often depicted as a shell model. The kernel of a UNIX system contains components like the scheduler and the *device drivers*. These routines permit access to the interface hardware and external devices. The memory management also resides in the kernel.

kernel

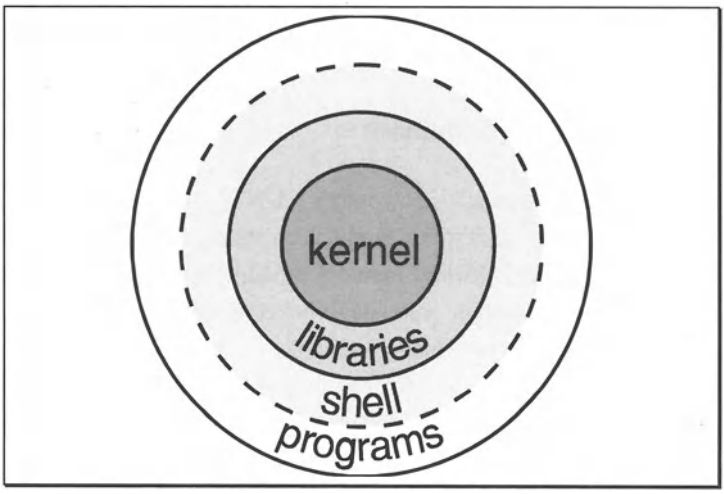


Fig. 2.1 Schematic structure of a UNIX system

The processes in the kernel are different from the processes running in the shell around the kernel. Normal user processes can be interrupted at any time, they are subject to the control of the schedulers, and a certain region of memory is allocated to each of them. If a user process attempts to access a region of memory outside its own, the process is aborted with the error message "segmentation fault".

By contrast, kernel processes have unrestricted access to all resources of the computer. Hence we speak of two different modes in which processes can run: the user mode and the kernel mode.

user mode/kernel mode

The outer shell of a UNIX system consists of programs that directly make contact with the user. This outer shell includes the command shell, which launches operating system commands, as well as application programs such as word processing and databases.

Between the outer shell and the kernel are the various libraries that provide access to their library functions (usually written in C) and to kernel routines. These libraries are normally linked to a program after its compilation, thus adding the library routines to those of the program itself.

libraries

Since statically linked programs demand a large amount of memory, modern programmers usually employ *shared libraries*, which consist of two parts. A small part containing only references to the library is linked to the program. The library itself is actually loaded only when the program is executed. Shared libraries allow multiple programs to use the routines in the libraries simultaneously, which also saves memory.

shared libraries

Another advantage is the possibility to exchange a shared library for a newer version without having to relink the programs relying on it. However, this assumes that the routines in the new library are invocation-compatible with those of the old version.

2.5 File systems

A file system manages data stored on a hard disk. Although every computer system has such mechanisms, they can differ substantially.

hierarchical structure

Modern file systems have a hierarchical structure. The user can distribute files in various directories, making it easier to maintain an overview. The user accesses the files via the path. Unlike DOS, UNIX uses the slash (/) as delimiter within a path.

DOS addresses disk drives and the partitions of a hard disk with a letter. UNIX merges all these to a single file system and does not assign them a separate designation. This means that the user can no longer distinguish the individual drives and partitions. Only one large drive with a single file system appears to exist. Problems arise in the management of diskettes and other removable storage media since these are not permanently in the drive. Before access to a removable storage medium is possible,

mount

it must be linked to the system using the `mount` command, which normally only the system administrator can do.

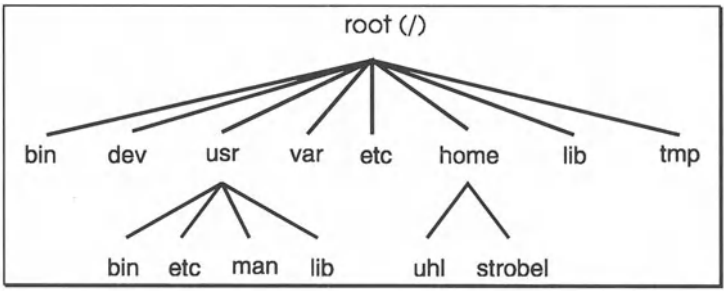


Fig. 2.2 UNIX file tree

Likewise the management of files and free blocks differs considerably under DOS and UNIX. DOS creates a File

FAT

Allocation Table (FAT) on each drive to record the free and allocated sectors. Another sector contains the root directory. Besides the names of the contained files, a DOS directory also contains their attributes, such as size and date.

By contrast, UNIX allocates an i-node for each file, in which the most important attributes are stored, such as name,

permissions and start block. Directories thus contain only references to the respective i-nodes. Because it is both more economical in terms of storage space and more efficient regarding access, such a structure proves better suited to the management of larger file systems then a FAT system.

Permissions

When a file is created under UNIX, the operating system stores not only the filename and the date of creation but also the user ID of its creator (or owner) as well as the group that owns the file. To be able to protect the files of a file system against undesired access, the permissions for each file are stored separately. Thus access to a file can be limited to its owner or to a certain user group. It is also possible to define general permissions. We further distinguish among read, write and execute permissions, which are reflected in the output of the `ls` (list) command by the letters `r`, `w` and `x`; the position of the letters in the output indicates whether the permissions apply to the owner of the file, the group owners of the file, or everyone else.

owner, group

ls

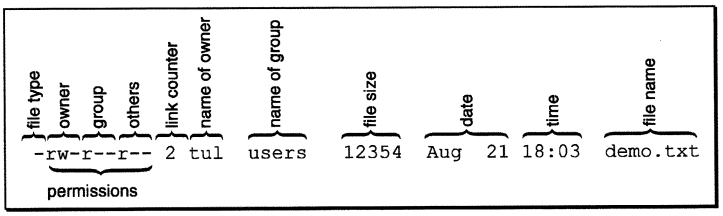


Fig. 2.3 Displaying permissions using the command `ls`

Subdirectories are an exception. Read permission suffices only to read their contents. In order to change to a subdirectory, however, a user must possess both read and execute permissions for that subdirectory.

directories

Links

Another feature of the UNIX file systems is the possibility to create links. If access to a file is to take place from various points in the file systems, this file could simply be copied. Naturally this approach would mean a waste of storage. In such cases under UNIX the creation of a *link* proves a more practical alternative.

UNIX links to a file can be either hard or symbolic links. A

hard link

hard link is an additional reference from a directory to a file or its i-node. The link counter tabulates the number of such references. If a file to which several links refer is to be deleted, then the link counter is decremented by one. Only after the link counter reaches the value one can the file be deleted physically.

Since the numbers of i-nodes are unambiguous only within a file system, hard links cannot be created for files intended for use

symbolic link

outside a given file system. By contrast, symbolic links can reference any directory entries (subdirectories or files). Whether the referenced file actually exists plays no role in the creation of the symbolic link.

The output of the `ls` command indicates the difference between these two kinds of links:

```
linux1:/etc>ls -l hosts passwd
lrwxrwxrwx 1 tul  users   10 Aug 21 18:05 hosts -> /etc/hosts
-rw-r--r-- 2 root root   863 Aug  9 15:00 passwd
linux1:/etc>
```

Symbolic links are represented with an "`l`" (el for link) in the column for the file type and a visible reference to the original file. Hard links can only be recognized by the increased link counter in the second column (after the permissions).

Virtual file system

In order to support the development of different file systems, Linux provides an additional layer, the *virtual file system*, between the kernel and the actual routines of a file systems, as in proprietary UNIX systems. The virtual file system defines a number of routines that must be available in every file system, for

opening, reading, writing and closing files. This unambiguous interface enables the problem-free coexistence of different file systems.

2.6 Devices

UNIX maps hard disks as well as terminals and other devices onto special files in the directory `/dev` of the file systems. Thus the programmer can access such devices in the same way as normal files.

/dev

This file-like view of devices also has advantages for the user. If the user wants to output a file to the printer instead of to the console, then it suffices to redirect standard output to the respective device (`/dev/lp1`):

```
linux2:/home> cat output.txt >/dev/lp1
```

Likewise the floppy disk drive (`/dev/fd0`), the mouse (`/dev/mouse`), and the hard disk are addressed via an entry in the `/dev` directory.

`/dev/console`	system console
`/dev/mouse`	serial mouse
`/dev/hda`	first IDE hard disk
`/dev/hda1`	first partition of first IDE hard disk
`/dev/hda2`	second partition of first IDE hard disk
`/dev/hdb`	second IDE hard disk
`/dev/hdb1`	first partition of second IDE hard disk
`/dev/sda`	first SCSI hard disk
`/dev/sdb`	second SCSI hard disk
`/dev/lp0`	first printer port (LPT1)
`/dev/null`	null device (all output is suppressed)
`/dev/ttyN`	virtual console
`/dev/ptyN`	pseudoterminal for login from network
`/dev/ttySN`	serial port

List of the most important device drivers

Files in directory /dev and the respective devices are associated via two numbers, the major device number and the minor device number. These also form the interface to the kernel. In addition, there are two kinds of devices, *character* and *block* devices. The former are character-oriented and are used primarily for devices like terminals and serial ports. The latter are used for transferring large data blocks for hard disks and other data storage devices.

The following ls output shows pseudofiles in the directory /dev.

```
linux1:/dev>ls -l
                                    ...
brw-r-----   1 root     root       3,   0 Aug 29  1992 hda
brw-r-----   1 root     root       3,   1 Aug 29  1992 hda1
brw-r-----   1 root     root       3,   2 Aug 29  1992 hda2
                                    ...
crw-rw-rw-   1 root     root       4,   0 Aug 16 12:26 tty0
crw--w--w-   1 tul      users      4,   1 Aug 21 15:15 tty1
                                    ...
linux1:/dev>
```

The major device number (fifth column in the above terminal capture) specifies the type of the device. Normally each major device number has a corresponding device driver in the kernel. When multiple devices of the same type are connected, they are distinguished by their minor device number (sixth column in the above figure) and served by the same driver.

2.7 Shells

UNIX commands such as ls, cd and man, which are entered in the normal command line, are not executed by the UNIX operating system itself; instead, they are programs that are usually located in the directories /bin and /usr/bin. A user entering such commands is not in direct contact with the UNIX operating system, but only with its outermost shell.

General background

We can compare the tasks of the shell with those of command.com under DOS, although a UNIX shell affords

significantly more features than the DOS command interpreter. The shell has few commands of its own and spends most of the time reading user input from the standard input device, i.e. the keyboard, and starting the respective program. With the input of `ls -l`, for example, the shell invokes the program `/bin/ls` with the option `-l`, and the contents of the current directory are displayed on the console.

Among the features of a shell are the redirection of input and output, the linking of multiple commands in a *pipe*, and execution of shell scripts. A shell script consists of several commands in the form that they would be executed interactively via the command line. Pipes allow channeling the output of one command to the input of a second command. This feature is often used to display long screen output pagewise by means of the command `more`. This is an example of using a pipe:

redirection

```
linux1:/home/tul> ls -l | more
```

Since a shell is itself an ordinary program that is started when a user logs in (login shell), the shell can be replaced with other versions.

Standard shells

Most proprietary UNIX systems include three kinds of shells: a Bourne shell (`sh`), a Korn shell (`ksh`) and a C shell (`csh`). The Bourne shell was the first UNIX shell; thus it affords little in the way of comfort. The Korn shell is an extension of the Bourne shell and appears relatively often with proprietary UNIX systems.

sh, ksh, csh

Like BSD UNIX, the C shell was born at the University of California at Berkeley. In contrast to the Bourne and Korn shells, the C shell offers a simple history function, allowing the UNIX user to return to commands that have been entered on the command line. *Alias substitution* makes it possible to automatically replace certain commands that were entered in an initialization file or interactively. Due to its C-like syntax, this shell variant has been a particular favorite among programmers.

When it is started as login shell, the C shell executes the commands of the file `.login`, which must be located in the home directory for this purpose. Since a user normally possesses only one active login shell, the contained commands are executed only once at login.

.cshrc

On the other hand, a shell script that is executed at the start of each new C shell is the file `.cshrc`, which likewise must be located in the home directory. This shell script often sets paths, defines environment variables, and declares alias substitutions.

alias

An alias is an additional name for a command for which options can be specified. The following excerpt from a `.cshrc` file declares the alias `l` for the command `ls` with the option `-lF`. This option appends a slash (`/`) to the display of directories and an asterisk (*) to executable files.

The following is an excerpt from a `.cshrc` file.

```
PS1='\h:$PWD>'
alias l "ls -lF"
alias ls "ls -F"
alias rm "rm -i"
```

The alias for `rm` causes the command to ask for confirmation from the user before the actual deletion of a file. Since this behavior can be annoying when deleting multiple files, it is possible to either invoke the `rm` command directly by prefixing its path (`bin/rm`) to the command, or to delete the alias with the command `unalias rm`.

The following shows the effect of an alias for ls and how to circumvent the alias by using the path.

```
dirk1:/home/stefan# ls
Mail/          faqxp4.apr     test*
dirk1:/home/stefan# /usr/bin/ls
Mail           faqxp4.apr     test
dirk1:/home/stefan#
```

alternative shells
bash and tcsh

As a rule Linux uses enhanced variants of the above shells rather than the originals. These more comfortable shell variants, which are available for free for almost all UNIX systems, render the original shells obsolete. The Bourne Again shell (`bash`) has

displaced the Bourne shell, and the `tcsh` shell has supplanted the C shell.

2.8 Daemons

background

Daemons are special processes that run in the background and usually assume important tasks in a UNIX system. With daemons, large parts of the operating system run as independent programs. This keeps the operating system kernel relatively small. Furthermore, individual daemons can be activated, or restarted after a change in configuration, even during operation. Since daemons run as independent processes, they can run in parallel with one another and thus do not block other programs.

Printer daemon (lpd)

At regular intervals the line printer daemon (`lpd`) checks the directory `/var/spool` for new printing jobs and outputs any such jobs to the respective printer. For outputting a file on a printer, Linux provides the `lpr` command known from BSD UNIX. New print jobs are normally appended to the end of the queue before they are output by the printer daemon.

BSD

Cron daemon

If a user wants to execute a program at certain times or at regular intervals, the cron daemon makes this possible. For each user, this daemon manages a table in which the times are entered when the desired processes are to start. The output of an executed command or the respective error messages are sent to the user as mail. If a script is to be executed only once at a certain time, the command `at` does the job. Repeated execution of a process at regular intervals requires an entry in the user's cron daemon table (`crontab`). A separate command named `crontab` serves this purpose.

crontab

Internet daemon (inetd)

Most Internet services have also been implemented as daemons. These daemons are activated only when there is actual demand for such services. If they were always active in the background like other daemons, they would unnecessarily consume memory and computation time. Therefore these daemons are not automatically started with the booting of the system.

port number

A UNIX system normally provides an *Internet daemon* (Internet superserver), which waits for messages from the network. Every service has a fixed port number. The file (`/etc/services`) contains a registry of services and their respective port numbers. Another file (`/etc/inetd.conf`) lists the respective daemons that offer the requested service. Only when an actual connection is requested does `inetd` start the daemon that takes over the connection. Upon termination of the connection, the daemon is also ended, while `inetd` continues to run.

Syslog daemon

daemon output

Since a daemon normally does not send output to the console, a separate protocol daemon was created to handle output and error messages from other daemons. This output can be displayed on the console, written to a file, or forwarded as a mail to the system administrator.

Network daemons

Most Internet services such as `ftp`, `telnet` and `mail` were realized with separate daemons that are activated by `inetd` on demand. The following synopsis gives an overview of the most important network daemons available under Linux.

- **`bootpd`** - needed for booting diskless workstations and X-terminals.
- **`fingerd`** - allows checking (`finger`) which users are active on (another) system.

- **ftpd** - for file transfer via `ftp` from one system to another.
- **mountd** - hooks a file system of another computer to the local file tree.
- **nfsd** - provides data as an NFS server.
- **nntpd** - transfers news from USENET.
- **rlogind** - permits remote login from another (remote) system using the `rlogin` commands.
- **rshd** - permits the execution of a command from another (remote) system.
- **smail** - sends and receives mail on the network. However, even without `inetd`, it can be important, for example in a UUCP configuration.
- **talkd** - permits interactive communication with other users via the command `talk`.
- **telnetd** - similar to `rlogind`, permits a user a remote login from another computer.
- **tftpd** - like `bootpd`, used to boot a remote machine on the network.

2.9 Overview of commands

To help newcomers to get started with Linux, we provide a collection of the most important UNIX commands along with brief explanations. More detailed information is available in the standard UNIX literature or the on-line reference manual.

- **ls** - outputs a list of files and directories. Optionally, the file sizes, corresponding permissions, and file owners can be displayed. Recursive output of complete directory trees is also possible.
- **cd** - changes to another directory. If no parameter is specified, the current directory becomes the home directory.
- **cp** - copies the specified files from one directory to another directory or to another file. Optionally, a complete directory tree can be copied recursively.
- **mv** - moves a file within a file system or renames a file or a directory.

- `rm` - removes a file. Optionally, an entire file tree can be deleted recursively.
- `mkdir` - creates a new directory.
- `rmdir` - removes an empty directory.
- `exit` - exits the current shell.
- `more` - displays the contents of a text file pagewise on the screen. In addition, character strings can be searched for in the file.
- `man` - displays the On-line Documentation (Manual pages).
- `cat` - intended for the concatenation (appending) of text files, but can also be used to output a file.
- `grep` - searches within the specified files for a certain pattern.
- `passwd` - changes a user's password.
- `ps` - lists all running processes with their process ID.
- `kill` - terminates the process with the specified process ID by sending a signal to the process.
- `su` - temporarily changes the user ID, without having to repeat the login. If "`-`" is specified as an additional parameter, this creates a renewed login.

Networking

A significant aspect in the discussion of modern workstations and their operating systems centers around networking capability, that is, the possibility to integrate the system into an existing network.

The complete development of Linux would have been impossible without the Internet. Thus even very early versions of the kernel included TCP/IP and the necessary drivers for PC network boards.

This chapter presents the fundamentals of networking and describes the programs that allow Linux and other UNIX variants to access network services.

3.1 Network hardware

Connecting to a network requires little in the way of hardware. Two computers can be connected in an elementary local area network (LAN) with two simple Ethernet boards, a piece of thin Ethernet cable, two T-connectors and two terminal resistors. This additional hardware costs less than $200, which makes networking affordable even for private users.

LAN

SLIP (Serial Line Interface Protocol) provides a more economical way to make a network connection. SLIP enables a TCP/IP connection via a serial cable or a modem. The other end of such a SLIP connection is usually a larger workstation or a SLIP router that forwards the IP packets to Ethernet. Such a network connection has found popularity especially among

SLIP

students and university employees who have access to a SLIP router and can thus connect via modem with the university network and the Internet.

CSLIP / PPP

Because the transfer of IP packets via SLIP is not particularly efficient, other protocols called CSLIP and PPP were created.

PLIP

The parallel port and a protocol called PLIP provide an economical TCP/IP connection. Two computers are connected with a null printer cable via free parallel ports. This type of network proves particularly attractive for data transfer between a notebook and a workstation.

3.2 TCP/IP

The de facto standard for networking UNIX computers is TCP/IP, a protocol that was developed for the Internet and has become available for almost all computer platforms. TCP/IP is not a standard like those of ANSI, ISO, and IEEE, but a manufacturer-independent definition that is available to everyone in the form of a Request for Comments (RFC).

RFC

An RFC is usually a description of a protocol or a proposal for a new protocol. RFCs can be found on many ftp servers as well as at the Network Information Center (NIC), e.g. `nic.ddn.mil`.

TCP/IP evolved in the early 1970s as a protocol for ARPANET (later DARPANET and then Internet), which at the time primarily interconnected American universities. The network was financed by the Advanced Research Project Agency. This U.S. government agency was primarily involved in military projects; hence it is no surprise that TCP/IP was also declared as the standard of the U.S. Department of Defense.

Meanwhile the Internet has grown to include many subnetworks worldwide.

Structure

TCP/IP essentially consists of four *levels* which, with some deviations, can be mapped to today's standard ISO/OSI reference model. The lowest level is termed the *network level* and

corresponds to *layers* one and two of the ISO/OSI reference model. In most cases Ethernet is used at this level. Although other constellations such as the token ring are possible, under Linux the only available drivers are for Ethernet boards, SLIP, CSLIP, PPP, and PLIP.

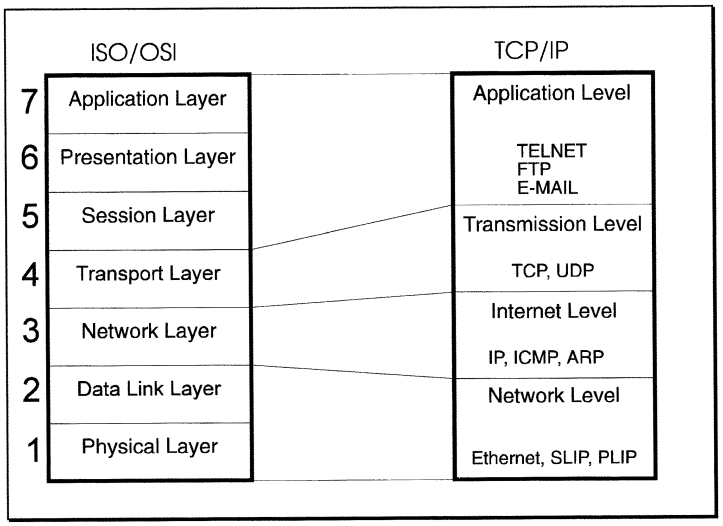


Fig. 3.1 Mapping of TCP/IP levels to ISO/OSI

IP

The second level is the *Internet level* with its Internet Protocol (IP). This level approximates the ISO/OSI layer three. This layer enables the transfer of data packets (datagrams) independently of the actual network that makes the connection. On this level, networking software identifies a computer by its unique IP address.

TCP and UDP

The third level corresponds to ISO/OSI layer four and offers TCP and UDP protocols. TCP stands for Transmission Control Protocol and makes a virtual connection, while UDP means User Datagram Protocol and offers a simple packet service.

port numbers

In addition to the network address, making a connection at this level also requires a port number, which usually corresponds to a particular network service (in `/etc/services`). This permits multiple independent connections via a single interface. Several of these port numbers are reserved for standard TCP/IP

applications: for example, port 23 is for telnet, and ports 20 and 21 are for ftp.

In order to provide a uniform interface for the programming of network applications, the developers of BSD UNIX introduced the *Berkeley sockets* in the early 1980s. These are a series of kernel routines that are needed to make a connection between two computers and for transferring data between them. Both TCP and UDP connections can be realized via sockets.

sockets

Note that it does not matter whether the data transfer takes place within a local area network (LAN) or in a global wide area network (WAN). The programming interface is always the same.

LAN/WAN

The Linux kernel also includes such a socket interface, which makes most well-known UNIX network applications available.

Applications

The fourth level of TCP/IP covers ISO/OSI layers five to seven and is termed the application level. While levels one to three are normally components of the operating system (in Linux they are part of the kernel), level four consists of normal programs.

The most important services on this level are telnet for the virtual terminal, ftp for transferring files, and e-mail for transferring electronic correspondence. Telnet and ftp exist as programs of the same name that the user can use directly. E-mail is supported from the TCP/IP end with only a simple transfer mechanism that is used by various other programs.

telnet, ftp, e-mail

To use ftp to establish a connection to the most important Linux ftp server in Finland, for example, the user inputs `ftp nic.funet.fi`. This starts the ftp program, which then attempts to establish a TCP connection to the remote computer on port number 21, which is used by the ftp daemon. Once the connection has been established, the user is prompted to enter the user ID. On ftp servers with public access, the user ID can be `ftp` or `anonymous` and the password should be the user's own e-mail address. The following shows an example of an ftp session:

ftp

```
linux2:/home/stefan>ftp sun1
Connected to sun1.
220 sun1 ftp server (SunOS 4.1) ready.
Name (sun1:stefan): ftp
331 Guest login ok, send ident as password.
Password (sun1:ftp): stefan@linux2.rz.fh-heilbronn.de
230 Guest login ok, access restrictions apply.
ftp> ls
200 PORT command successful.
150 ASCII data connection for /bin/ls (141.7.1.41,1157) (0
bytes).
total 6
drwxrwxrwx  2 0        150           512 Jul 16 16:14 Incoming
-rw-r--r--  1 0        1             139 Aug 22 12:33 README
drwxr-xr-x  2 0        150           512 May 10 09:51 bin
drwxr-xr-x  2 0        150           512 May 10 09:54 dev
drwxr-xr-x  5 0        150           512 Jun 20 15:36 pub
drwxr-xr-x  3 0        150           512 May 10 09:52 usr
226 ASCII Transfer complete.
ftp> get README
200 PORT command successful.
150 ASCII data connection for README (141.7.1.41,1158) (139
bytes).
226 ASCII Transfer complete.
142 bytes received in 0 seconds (0.14 Kbytes/s)
ftp> bye
221 Goodbye.
```

ftp commands

Then the user can employ the commands of the ftp program, such as `cd`, `dir`, `get`, or `put`, to find and transfer files. These are not the commands used in the ftp protocol; instead, the ftp program recognizes these commands and converts them to the associated protocol commands such as `PORT` or `RETR`. Only these protocol commands are transferred. For a data connection in the transfer of files, a second TCP connection is established.

The individual commands for the ftp program are available on the on-line reference manual page for `ftp` or from the help function within the ftp program, which is invoked by inputting `help`.

Telnet

To log in on another computer with telnet, the user invokes the telnet program with a computer name or an IP address. This establishes a TCP connection to the telnet daemon of the specified computer. This is an example of a telnet login:

```
linux2:/home/stefan>telnet sun1
Trying 141.7.1.20...
Connected to sun1.
Escape character is '^]'.


SunOS UNIX (sun1)

login:
```

The telnet program uses the port number of the telnet daemons only as a default value. Telnet can be invoked with an optional port number in addition to the host name. For example, `telnet <Hostname> 25` establishes a connection to port 25 of a computer that is reserved for SMTP, the e-mail-protocol. Specifying port number 119 could establish a connection to the news daemon of a news server.

This feature even goes so far that entering `help` often elicits descriptions of the protocol commands of the server's daemons. Such features prove a valuable aid in debugging.

MUDs and Chess Server

In addition, on certain ports some servers offer games such as text adventures, multi-user dungeons (MUDs), and chess. The only requirements are the name of the host and the respective port number. For example, an Internet chess server (ICS) is located on host `ics.uoknor.edu` on port number 5000. To use this server, establish a connection as follows:

```
stef1:/users/strobel>telnet ics.uoknor.edu 5000
Trying 129.15.10.21 ...
Connected to remus.ucs.uoknor.edu.
Escape character is '^]'.

     **** Welcome to the Internet Chess Server at ics.uoknor.edu ****

     This program was written and is maintained by Daniel Sleator
     <sleator@cs.cmu.edu> (aka Darooha). Direct general questions about the
     chess server -- as well as comments about bugs and features -- to him.

     Other people are in charge of various specific aspects of this system,
     such as: the local host, rules and etiquette, client programs (giics,
     xboard, etc), and registration. Type "help people" upon logging in to
     see who to contact for specific purposes.

Enter Login:
```

The program `xboard`, which normally serves as the graphical front end for the GNU chess program, is able to establish such a telnet connection and in parallel to graphically represent the positions.

list of services

Lists of sources for such Internet services are stored on various ftp servers under the name `internet.services` or `internet.resources`. Along with FAQs (frequently asked questions), such lists are published regularly in the newsgroup `news.answers`. Particularly newcomers to the Internet should learn to use a news reader, since this eases access to other servers and information sources (see Section 12.2).

The TCP/IP e-mail (electronic mail) service normally consists of a daemon that handles the transfer of correspondence to other computers using the SMTP protocol, and programs for writing and reading letters (e-mail).

sendmail and smail

Linux furnishes the widely propagated `sendmail` program as well as the alternative `smail`, which handles TCP/IP as well as UUCP. As a mail front end for reading and writing letters, most Linux packages provide `elm` and `pine`. In addition, there are graphical programs such MuMail for the comfortable management of e-mails under the X Window System (see also Section 12.1).

3.3 Berkeley r-utilities

TCP/IP

The University of California at Berkeley's UNIX exerted a great deal of influence on networking capability under UNIX. Through the successful integration of TCP/IP into the UNIX operating system, this implementation made a significant contribution to the further proliferation of TCP/IP.

remote

The Berkeley r-utilities are a group of programs whose names begin with the letter r, the most important being `rlogin`, `rsh` and `rcp`. The r stands for remote. The Berkeley r-utilities belong to the standard software of a networked UNIX workstation. Over time they have been enriched by a number of programs such as `rwho` and `ruptime`. As in many proprietary UNIX variants, utilities from the Berkeley 4.2 Distribution were ported to Linux.

trusted users

The basic idea of these utilities is to provide a user who has an account on multiple computers on a network, a simple way to log in to other computers on the network, to let programs run, or to copy files, without having to enter a password each time. To do this without creating a security flaw, certain users from other computers can be defined as *trusted users*. Only such users can then log in to the computer and use its services without having to enter a password. This definition is system wide in the file `/etc/hosts.equiv` or in `.rhosts` in the home directories of the respective users for their own accounts.

Reserved ports provide an additional security measure. Here the server checks the TCP port number of the client: if it is not a reserved port, the server denies the connection. Reserved ports on UNIX machines are available only for users with superuser access privileges. This prevents a normal user with another, self-written program from feigning a false computer or user name in order to gain access to another computer. However, this means that the r-utilities have to run with the user ID *root*.

Kerberos

Newer implementations of the Berkeley r-utilities also support Kerberos, a network authentication program based on a key distribution model. The Kerberos system allows users and servers to prove their identity. It was developed at the Massachusetts Institute of Technology (MIT). More information on Kerberos can be found in the Kerberos FAQ (see Section 8.4).

rlogin

remote login

For the user, the remote login program `rlogin` functions similarly to the `telnet` program, except that with an appropriate configuration it requires no user ID or password. `rlogin` does not permit the specification of a deviating port number.

```
stef1:/home/strobel>rlogin lia
Last login: Sun May 29 09:33:50 from STEF1
SunOS Release 4.1.2 (GENERIC-12) #1: Tue Apr 12 20:52:39 MET
DST 1994

LIA:/users/strobel>
```

rcp

remote copy

The `rcp` supports remote copying of files between computers. In contrast to normal ftp, `rcp` can also recursively copy subtrees of file hierarchies. However, the exact access path must be specified. Therefore in many cases users prefer the interactive ftp utility or access via a *network file system* (NFS) (see Section 3.4).

```
stef1:/home/strobel>rcp -r lia:~/emacs .
stef1:/home/strobel>
```

rsh

The remote shell `rsh` allows a user to execute programs on other computers. Beyond the command to be executed, the name of the remote computer and, optionally, a user ID are specified. The output of the executed command is then routed back to the local machine via the network.

remote shell

Beyond the remote execution of commands on other machines, this program is also adept at fast data transfer between computers. Here the user exploits the `rsh` feature that allows combining the local machine's own standard input and output with the program that it executes on the target computer. For example, this enables making a backup of files from a local hard disk onto a streamer on another computer on the network.

data transfer with rsh

Using GNU tar and gzip in combination with rsh makes it easy to realize compressed transmission. The following example demonstrates this.

```
stef1:/home/strobel>rsh lia "ls .em*"
.emacs
.emacs-bkmrks
.emacs-places
.emacs-places~
.emacs-skp
.emacs~
stef1:/home/strobel>
stef1:/home/strobel>rsh lia "tar cfz - .em*" | tar xvfz -
.emacs
.emacs-bkmrks
.emacs-places
.emacs-places~
.emacs-skp
.emacs~
stef1:/home/strobel>
```

3.4 NFS

The *network file system* (NFS) makes it possible to mount file systems that are released (exported) by other computers, as a part of the local machine's own file systems. This provides transparent access to the directories of the remote computers.

The following example shows access to files in the `/home` directory of a remote computer (`stef1`) on the network. Initially the directory `/stef1` on the machine `dirk1` was empty. After the mount procedure this directory effectively contains all files of the directory `/home` of computer `stef1`. The following is an example of this NFS `mount` procedure.

```
dirk1:/# ls
bin/          install/     lost+found/  stef1/       var@
dev/          lastlog@     mnt/         tmp/         vmlinux
etc/          lib/         proc/        user/        vmlinux.old
home/         linux@       root/        usr/
dirk1:/# ls stef1
dirk1:/# mount stef1:/home /stef1
dirk1:/# ls stef1
dirk/    fritz/   ftp/     peter/   root/    stefan/
dirk1:/#
```

Sun Microsystems

NFS was developed by Sun Microsystems. Because Sun released the definitions of the protocol, many other manufacturers were able to integrate NFS into their operating systems. Thus NFS became a standard that asserted itself on almost all platforms even though it was not controlled by any higher authority. NFS is available for almost all UNIX variants, DOS, and other operating systems.

stateless server

A significant feature of NFS is the *stateless server*. This means that the NFS server that exports directories does not store any state information on the clients, but only carries out simple read and write operations. For example, if an NFS server needs to be restarted for whatever reason while an NFS client is copying a file from a directory exported by the server, the client's copying procedure is not interrupted; instead, the client waits until the server responds again and then continues the copying procedure. However, this procedure becomes problematic when it comes to the synchronization of access by multiple clients who want to write to the same file.

security

Another drawback of NFS is the security of its protocol in terms of privacy and unauthorized access. Newer distributed file systems such as the Andrew File System (AFS), which OSF DCE uses, prove superior to NFS. However, the alternatives have not attained the level of proliferation that NFS enjoys.

Linux also provides NFS. However, the respective drivers must have been compiled into the kernel. Then at run time the `portmap` daemon as well as the `nfsd` and `mountd` daemons must be running to allow the use of NFS. The file `/etc/exports` on the server specifies which directories an NFS client can access. Additional information can be found in Chapter 6.

3.5 RPC

RPC

NFS is based on Sun Microsystems' *remote procedure calls* (RPC), a mechanism that allows execution of individual routines on a remote computer on the network. There is an RPC server that provides subroutines and clients that invoke these subroutines with parameters. RPC is also a special kind of communication between processes on a network; in contrast to network programming via sockets, RPC abstracts away from a communication channel.

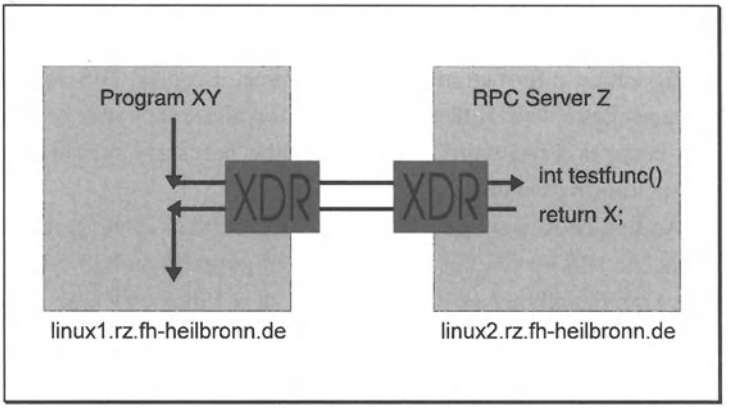


Fig. 3.2 RPC invocation

XDR

XDR is normally used along with RPC. This machine-independent method of data representation is used to exchange data between computers with different processor architectures. (One frequent difference between computers with different processor architectures is the representation of integers, for example.) The XDR description allows the definition of the

parameters of an RPC routine; then appropriate subroutines convert the parameters between the machine-independent format and each computer's own format.

The invocation of these conversion routines and of the internal RPC routines can be automated with the program `rpcgen`. It generates C source code for the RPC server and the client from a formal description of the RPC routines.

portmap

To enable the establishment of a connection to an RPC server on another machine, a `portmap` daemon must be running on the latter. This daemon "knows" which services are available and forwards the RPC invocations from the network to the correct RPC server. The information that the `portmap` daemon has stored about registered programs can be retrieved for diagnostic purposes using the `rpcinfo` program.

3.6 NIS

Since the consistent management of accounts and their passwords on a network can be quite arduous, Sun Microsystems developed the *Network Information System* (NIS). Instead of storing user information, available network services, and other system configuration information on each machine, a central NIS server manages these data. If the administrator creates a new user or if a user changes a password, NIS handles the necessary forwarding of this information to all affected machines.

NIS client

As this book went to press, a Linux computer could not yet run as the NIS server, but it could be configured as an NIS client. This significantly eases the integration of a Linux machine into an existing UNIX network.

The NIS client software is included in the Slackware distribution or can be drawn from most ftp servers, for example, from the directory `/pub/linux/local/yp` on the server `ftp.uni-paderborn.de` (University of Paderborn, Germany).

3.7 Other network services

As explained in Section 3.2, a dedicated ISO standard covers the structure and implementation of networks. Although TCP/IP can be mapped to the OSI model, it is not compatible with the standard. A developer who wants to create OSI-compatible applications under Linux can use the freely available ISO/OSI stack named ISODE (by Marshall T. Rose) which has been ported to Linux.

Novell networks

To provide connectivity to Novell networks (IPX protocol), there are several approaches to providing transparent access to netware drives. The corresponding drivers are still under development. Initial versions are already available, for example, on the ftp server `sunsite.unc.edu` in the directory `/pub/Linux/system/Network/sunacm/BETA`.

LAN Manager and WfW

The directory `/pub/tridge/samba` on the `ftp` server `nimbus.anu.edu.au` contains an SMB server and client named samba that can be used under Linux. This enables a Linux computer to function as server and client for IBM's LAN Manager and for Windows for Workgroups.

Linux Features

This chapter assumes that the reader has a basic knowledge of UNIX or has read the previous chapters. Here we describe in more detail some of the important characteristics and features of Linux that distinguish this system from other UNIX variants and from other PC operating systems.

4.1 Virtual consoles

Many PC UNIX implementations support *virtual consoles*, which provide the capability to manage multiple independent login sessions on one console. Switching between the individual sessions usually occurs via a special key combination.

multiple login sessions

Under Linux the `<Alt>` key combined with a function key enables switching between virtual consoles. The maximum number of virtual consoles is determined in the kernel. The file `/etc/inittab` establishes the configuration for which of these consoles is to display the login prompt.

Under the X Window System the `<Alt>` key is reserved for applications. The X Window System handles the switching to another virtual console with the three-way combination of `<Ctrl-Alt>` and the function key corresponding to the virtual console number. This allows the user to switch between the graphical interface of X Window System and the text-oriented interface of the virtual consoles.

X Window System

It is even possible to start multiple X servers. However, this is not recommended because there is seldom enough memory and thus performance suffers noticeably. Instead, use a virtual

window manager under X Window System, such as `olvwm` or `fvwm`, which also offers multiple virtual desktops under X11.

4.2 Linux file systems

The multitude of available file systems under Linux might seem confusing at first glance. The following subsections list these file systems and describe their most important features.

MINIX file system

1st file system

The first Linux versions furnished only one type of file system, and it relied heavily on the MINIX file system. This circumvented the effort required for a completely new development. Furthermore, this provided a stable file system from the start, although the MINIX file system certainly has some significant drawbacks as well.

limitations

File names cannot exceed 14 characters in length and the size of a partition is restricted to 64 MB. Although newer versions of this Linux/MINIX file system permit longer file names (30 characters), this file system is scarcely in use today.

However, it is noteworthy that, in contrast to many proprietary System V implementations, even this first version of a Linux file system supported symbolic links.

Extended file system (ext)

To overcome the above restrictions, Remy Card implemented the first alternative file system. His Extended File System (`ext`) for the first time supported files and partitions of up to 4 GB. He also raised the maximum length of file names to 255 characters.

partition size and longer file names

But this file system also has its weaknesses. Free blocks and i-nodes are not managed in a bit vector, but in a linked list. After a longer period of operation, this leads to extensive fragmentation of the memory, which causes noticeably longer access times.

fragmentation

Extended2 file system (ext2)

widespread file system

From the Extended File System, the Extended2 file system evolved after some time; this is currently the most widespread file system under Linux. The fragmentation problems no longer occur in this system. Furthermore, Extended2 supports a mechanism that saves lost sectors in a special directory. A possible system crash and its resulting corrupt file system are recognized when the system is started, and the damage can be repaired with a special utility (`e2fsck`).

Due to the flexible structure of the Extended2 file system, future versions will likely enable partitions of up to 4 terabytes.

Xia file system

alternatives to ext2

The Extended2 file system has not been the only attempt to establish a new, faster file system. At nearly the same time the Xia file system, named after its author, Frank Xia, appeared. This file system also increased the maximum partition size to 4 GB. File names can be up to 248 characters long. The size of a file is currently limited to 64 MB.

Other file systems

DOS & OS/2

The DOS file system permits Linux transparent access to DOS diskettes or partitions (and OS/2 FAT partitions). This permits access to pre-existing data, as explained below.

System V

To access Xenix, System V, and OS/2-HPFS partitions, special file systems were developed, but some of these are not yet fully implemented.

ISO 9660/HighSierra file system

CD-ROM

To provide access to CD-ROMs, Linux provides both ISO 9660 and High Sierra file systems. Likewise the Rockridge Extensions supporting longer file names have been implemented.

Proc file system

kernel and process information

The proc file system does not manage physical files, but enables access to data in the kernel and the currently running processes. On system startup the proc file system is normally mounted to the subdirectory `/proc` in the root directory. For every running process, this directory contains a subdirectory whose name is the respective process ID. The files that it contains provide a flexible interface to the actual process-specific information. In general these are virtual ASCII files whose contents can be output with the `cat` command. This allows the user to determine the contents of the command line or the environment variables of a process. Information about memory requirements, the parent process or the current process state can be extracted in this way.

4.3 Data exchange

Linux is seldom the sole operating system on a PC. More frequently another partition or hard disk contains DOS with MS-Windows, OS/2 or another PC UNIX variant. A user switching from DOS to Linux is seldom willing to sacrifice the old programs.

The following subsections show how Linux is able to work with other operating systems and exchange data and programs with them.

boot manager

For using different operating systems on one computer, a boot manager proves essential. At system startup, this utility enables the user to select the operating system to be booted. The Linux Loader (LILO) fulfills this function, among others. Chapter 5 describes the installation and operation of LILO.

MTools

DOS files

Since particularly the exchange of files with a DOS system is required nowadays of nearly all operating systems, for some time there have been freely available programs for processing DOS files under UNIX. As with many other UNIX systems, Linux provides the MTools for this purpose. These are commands like

mdir and mcopy that support the reading of the directory of a DOS storage medium (typically a diskette) and the copying of files. The following is an example of accessing a DOS diskette with MTools:

```
dirk1:/home/stefan# mdir a:
 Volume in drive A is dosdisk1
 Directory for A:/

COMMAND  COM       55591    3-10-93   6:00a
WINA20   386        9349    6-11-91  12:00p
AUTOEXEC BAT         359    8-26-93   9:02p
CONFIG   SYS         377    5-23-93   2:48p
DOSKEY   COM        6012    6-11-91  12:00p
EDIT     COM         429    6-11-91  12:00p
FORMAT   COM       34223    6-11-91  12:00p
       9 file(s)     1270784 bytes free
dirk1:/home/stefan# mcopy -t a:autoexec.bat .
Copying AUTOEXEC.BAT
dirk1:/home/stefan# mdel a:autoexec.bat
dirk1:/home/stefan#
```

DOS file system

In addition to MTools, Linux provides another method to access DOS storage media, the DOS file system. This makes it possible to mount diskettes and DOS partitions of a hard disk in the same way as other file systems in the Linux directory tree. This provides completely transparent access to the contained files. Access is faster than with MTools because the input and output operation can now profit from the cache of the operating system. The following is an example of accessing a DOS partition with the DOS file system:

mount DOS diskettes and partitions

```
dirk1:/# cd msdos
dirk1:/msdos# ls -a
./   ../
dirk1:/msdos# cd ..
dirk1:/# mount -t msdos /dev/hda2 /msdos
dirk1:/# cd msdos
dirk1:/msdos# ls -a
./              command.com*   format.com*    tools/
../             config.sys*    io.sys*        wina20.386*
autoexec.bat*   dos/           msdos.sys*     windows/
dirk1:/msdos# cd dos
dirk1:/msdos/dos#
```

However, the drawback of mounting diskettes is that they can no longer be inserted and removed at will; instead, the commands mount and umount must be invoked with each change of

diskettes. If the diskette is removed while it is still mounted, then the diskette inserted subsequently is usually overwritten.

access privileges & DOS file system

Since DOS provides neither user IDs nor group IDs, the individual files of a mounted DOS file system cannot be assigned individual settings for access control. However, on mounting the volume, group and user IDs as well as access privileges for the overall file system can be specified as an option. This at least permits control of access to the file system as a whole.

A complete description of how the DOS file system functions under Linux and the available options can be found in the on-line reference manual page for `mount` and the file `README.dosfs`, which resides on ftp servers along with the source code of the file system. There are two reference manual pages for `mount`, one for the command and one for the routine in the C library. To display the correct on-line reference manual page, the section must be specified in the invocation of the `man` command. The invocation for the `man` command would be:

```
stef:/# man 8 mount
```

The most important options that can be specified in the invocation of the `mount` command for a DOS file system are:

- `uid=<number>`
- `gid=<number>`
- `umask=<number>`
- `conv=binary` or `text` or `auto` (see next paragraph)

The following example shows the mounting of a DOS file system with the specification of options:

```
stef1:/# mount -t msdos -o umask=000 /dev/hda1 /dosc
```

The options can also be specified in the file `/etc/fstab`:

```
dev/hda3        none      swap        defaults
/dev/hda2       /         ext2        defaults
/dev/hda1       /dosc     msdos       rw,umask=000
none            /proc     proc        defaults
```

Text conversion

A fundamental problem in data exchange between DOS and UNIX is their different representations of new lines in text files with respect to carriage return (CR) and line feed (LF). In `mcopy` this has been solved with the command line option `-t`, which specifies whether to copy a file in binary mode or in text mode with conversion.

new line (CR/LF)

Since a mounted DOS file system allows access to many different files, there is no universal solution. Although there are options for the `mount` command that activate automatic conversion, this process does not always function reliably. Whether a file is a text file or a binary file cannot always be decided with certainty.

automatic conversion

4.4 Emulators

For many users the exchange of data between Linux and other operating systems alone does not suffice. They also need access to programs written for DOS, MS-Windows, or other UNIX systems. The following subsections describe how Linux provides support in this area.

DOS emulator

Like OS/2 and other newer operating systems, Linux provides a DOS emulator to allow DOS programs to run simultaneously with other Linux applications. However, this emulator does not copy the entire DOS operating system, but only provides the rudimentary input/output routines (BIOS) that enable access to important devices.

The DOS emulator system allows booting DOS under Linux. The specific DOS version (MS-DOS, PC-DOS, DR-DOS, Novell DOS, version number) is of secondary importance.

graphics, HMA, UMB, EMS and DPMI

Overview. Although the emulator is still far from being able to support every DOS program without problems, its capabilities are impressive. Programs can be run in graphic mode and with the support of high memory, upper memory blocks (UMB) and expanded memory (EMS). Support of DOS Protected Mode Interface (DPMI) and access to Novell servers from the emulator are currently under development. Programs such as Norton Commander and the text editor QEdit run problem-free, and larger commercial products such as Turbo Pascal and WordPerfect can also be used.

Since the user can switch between virtual consoles even within the emulator, under Linux the switching can take place among several DOS and UNIX applications. As under X11, switching is initiated with the key combination <Ctrl> and <Alt> simultaneously with a function key.

The following shows the output of the DOS emulator on booting DOS.

```
 1 MB display memory

Linux DOS emulator 0.49 $Date: 1993/05/04 05:29:22 $
Last configured at Sat Jul 31 18:40:22 1993
                 on softland, Linux 0.99.11 #3 Sat Jul 24
08:48:18 GMT 1993
maintained by Robert Sanders, gt8134b@prism.gatech.edu

[Linux file system] drive C: id directory /msdos/
Welcome to the Linux DOS emulator version 0.49!
A:\>
A:\>mem


    655360 bytes total conventional memory
    655360 bytes available for DOS
    638448 bytes max. size of executable program

   1024000 bytes total continuous extended memory
         0 bytes continuous extended memory available
   1024000 bytes XMS memory available
           DOS resident in High Memory Area
```

disk images

Besides normal diskettes and DOS partitions, the DOS emulator can also access floppy disk images and hard disk

images. For DOS programs, these files behave like real storage media. They are used primarily to boot the DOS emulator.

The DOS emulator is configured by editing a configuration file. Earlier versions of the DOS emulator used the directory `/etc/dosemu/` for all files, including image files, and their configuration file was `/etc/dosemu/config`. However, this contradicts the Linux file system standard. Thus newer versions use the configuration file `/etc/dosemu.conf` and place the image files in `/usr/lib/dosemu` or `/var/lib/dosemu`.

file system standard

Booting the emulator. The DOS emulator simulates only the hardware and the BIOS of a PC. Thus starting the emulator requires a DOS boot diskette, a hard disk partition with DOS installed, or a corresponding image file.

real DOS for booting

Directly accessing the DOS partition hard disk and booting from this partition provides one possibility. In this case, however, this partition must not be mounted in the Linux file system, since this could cause conflicts that could result in lost data.

The alternative is to boot from a diskette or from a special image file that the DOS emulator uses like a real diskette or hard disk. Although using a file instead of a boot diskette offers an elegant and quick solution, this image file must be created first.

booting from image file

Creating the boot image file for booting requires a normal bootable DOS diskette. Several special DOS programs that are included with the emulator also need to be copied onto this diskette. These programs allow access to the Linux file system from within the emulator and permit certain settings of the emulator to be read or modified. We also recommend copying a simple editor onto the diskette.

creating an image file

The following terminal capture shows the subdirectories of the DOS emulator distribution containing the driver and programs:

```
stef1:strobel/dosemu0.50pl1/commands>ls
Makefile       dosdbg.c       exitemu.S      lredir.exe     vgaon.S
bootoff.S      dosdbg.exe     exitemu.com    lredir.readme  vgaon.com
bootoff.com    dosdbg.readme  lancheck.exe   pdipx.com
booton.S       dumpconf.asm   linpkt/        vgaoff.S
booton.com     dumpconf.exe   lredir.c       vgaoff.com
stef1:strobel/dosemu0.50pl1/commands>
```

If these files are not included in your distribution, then the complete emulator distribution can be copied from one of the usual ftp servers. On the server `sunsite.unc.edu`, for example, the DOS emulator distribution resides in the directory `/pub/Linux/system/emulators/dosemu`. To copy the files from Linux to a DOS diskette, use MTools or mount the diskette.

copying the boot disk to a file

After all programs and files have been copied to the boot diskette, it can be copied to a file using the command `dd`. This file can be stored in any directory. If it is for personal use, employ the home directory; if multiple users will be using the DOS boot image, a system subdirectory should be created, such as `/var/lib/dosemu`.

Here we copy a diskette to the disk image file `bdisk`:

```
# dd if=/dev/fd0 of=bdisk bs=16k
```

Now the configuration file for the DOS emulator must be modified so that this image file can be used for booting. The corresponding entries in the configuration file are:

```
boota
bootdisk { heads 2 sectors 18 tracks 80 threeinch file bdisk}
```

Unless the image file is located in the current directory, its complete path must be specified in the configuration file.

Theoretically the DOS emulator could be started now with the entry of `dos`. For reasonable utilization, however, we still need to make some settings.

Accessing DOS Partitions. After the boot disk image has been created as described above, the DOS emulator can be booted, but there is no access to the Linux file system or to a DOS partition on the hard disk.

lredir and emufs.sys

To access the Linux file system from within the emulator, use either the driver `emufs.sys` or the program `lredir`. The driver `emufs.sys` can map any directory of the Linux file system onto the next free drive letter. In addition, the redirector program

`lredir` can even assign drive letters. For example, `lredir` is invoked as follows :

```
lredir c: \linux\fs/
```

The first parameter, in our example `C:`, specifies the drive letter to be used. The second parameter identifies the source, in our example the root directory of the Linux file system. `\linux\fs` represents the Linux file system and `/` specifies the path within the file system. The `lredir.readme` file in the same directory as `lredir` program itself in the DOS emulator distribution contains a detailed description of all the program's options.

other ways to boot

There are many other ways to boot from a disk image, but they all have consequences for access to floppy disk drives or other drive letters. We offer a brief overview of these approaches.

Assume that the hard disk contains a DOS partition that was mounted under Linux as directory `/dosc`. There are several ways to access this partition under the DOS emulator:

boot from diskette

- Boot from a diskette and invoke the driver `emufs.sys` in the file `config.sys` or the program `lredir` in `autoexec.bat`. This makes a directory under Linux the next alphabetical drive in the DOS emulator. In this case this would be `C:`. Since it is rather troublesome to continually insert a diskette during the booting of the emulator, this method is used seldom and usually only during the installation.

boot from HD image

- Boot from a hard disk image and use `emufs.sys` or `lredir` as above to access drive `D:`. The hard disk image file uses `C:` in this case.
- Boot from a hard disk image and use `lredir` in the file `autoexec.bat` to replace the drive letter C:. This makes the DOS partition accessible as `C:` and the floppy disk remains untouched. This provides the same environment as booting from DOS.

  The problem with this method is that the "hard disk" from which the file `autoexec.bat` is read is replaced by another

while the file is still being read and executed. Thus the file should be identical in the hard disk image and in the real hard disk partition.

- Activate a disk image file as drive A: and use the file `config.sys` or `autoexec.bat` as described above. Here the advantage is that no diskette is needed to start the DOS emulator. The drawback is that the disk drive can no longer be used as drive A: because this letter is used by the disk image.
- Use the special option `bootdisk` (see the start of this section) in the configuration file of the DOS emulator to boot from a disk image and switch this image off at the end of the file `autoexec.bat`. This is the most elegant method. Booting produces a normal environment without image files, yet a disk image can be used for booting.

DANGER

Caution! It is theoretically possible to access a DOS partition on a hard disk both directly via the DOS emulator and indirectly via the Linux file system, assuming that the DOS file system is mounted. To prevent data loss, avoid this dual access in any case. To access a DOS partition from both Linux and the emulator, the emulator should access the DOS partition not directly, but via the device driver `emufs.sys` or `lredir` and the Linux file system.

Configuration. The following example shows a complete configuration file for the DOS emulator in which the option `bootdisk` is used as described above. Lines beginning with # are comments.

```
# Example of a configuration file for the DOS emulator

# No debug messages
debug -a

# DOSemu Version message on startup
dosbanner on

# Enable access to video board ports
allowvideoportaccess on

# Keyboard and timer interrupts
keybint on
timint on

# Mouse options :
#     "microsoft"(default), "mousesystems3", "mousesystems5",
#     "mmseries" or "logitech".
serial { mouse microsoft device /dev/cua0 }

#serial { mouse device /dev/cua0 }
#serial { mouse device /dev/cua1 }

# IPX in DOS-emulator (under development)
ipxsupport off

#video { vga console }
#video { vga console graphics chipset et4000 memsize 1024 }
#video { vga console graphics chipset trident memsize 1024 }
#video { vga console graphics chipset diamond }

# No special supported video board, but graphic mode is ok.
video { vga console graphics }

# Direct access to keyboard
RawKeyboard

mathco off          # on or off
cpu 80386
boota               # booting from A:
xms 1024            # XMS size in K, or off
ems 1024            # EMS size in K, or off

# Here certain IO ports can be allocated to the DOSemulator
# ports { 0x388 0x389 }
# ports { 0x2f8 0x2f9 0x2fa 0x2fb 0x2fc 0x2fd 0x2fe }
# ports { 0x21e 0x22e 0x23e 0x24e 0x25e 0x26e 0x27e 0x28e 0x29e
}

# Access to speaker
speaker native          # native, off or emulated

# Direct access to hard disks or hard disk  images is set.
# In our example we use lredir to access the partition
# /dev/hda1 --- no direct access

#disk { image "/usr/lib/dosemu/hdimage" }
#disk { partition "/dev/hda1" 1 readonly }
#disk { wholedisk "/dev/hda" }

# The file bdisk is used for booting. It is created  from a
# boot diskette with the following command:
# dd if=/dev/fd0 of=bdisk bs=16k
# At the end of the file autoexec.bat for this boot file,
# the file is disabled with the command bootoff and so access
# to drive A is restored.

bootdisk { heads 2 sectors 18 tracks 80 threeinch file bdisk }
```

```
# Direct access to floppy disk drives
floppy { device /dev/fd0 threeinch }
floppy { device /dev/fd1 fiveinch }

# This example uses no lasting disk image.
#floppy { heads 2 sectors 18 tracks 80 threeinch file bdisk }

#Printer access is not activated in this example.
#printer { options "%s" command "lpr" timeout 20 }
#printer { options "-p %s" command "lpr" timeout 10 }
# pr format it
#printer { file "lpt3" }
```

The file `autoexec.bat` used in the boot disk image could look like this:

```
@echo off
lredir c: linux\fs/dosc
PATH C:\BAT;C:\TOOLS;C:\DOS;C:\EMU;C:\WINDOWS\WIN31;

lh DosKey
prompt $p$g
c:
bootoff
```

After `autoexec.bat` has run, the boot file is no longer accessible because the command `bootoff` disables the disk image and restores access to the real floppy disk drive. To modify the boot configuration afterwards, the bootfile has to be enabled again with `booton`.

changing boot configuration

Note that no blank lines appear in the file `autoexec.bat` after the command `bootoff`; otherwise DOS would attempt to continue after the command `bootoff`. Since the file `autoexec.bat` no longer exists at that time, an error message would result.

exiting DOS emulator

To end the DOS emulator, the program `exitemu` can be invoked, the key combination `<Ctrl-Alt-PageDown>` can be pressed, or the DOS emulator process can be terminated from another virtual console with the command `kill`.

X11 DOS emulator

Besides the DOS emulator itself, which runs on a console in text or graphic mode, there is a second variant that permits running a DOS program in an X window. By redirecting output to another

computer in the network, it also becomes possible to use DOS programs that run under the DOS emulator on other UNIX workstations.

Since a variant of the Color XTerm program is used to represent screen output, the X11 DOS emulator also offers a scrollable history. In addition, the mouse can be used to copy screen output from one window and paste it in another window.

Unfortunately the X11 DOS emulator does not support DOS programs that run in graphic mode. In this case the user would have to resort to the normal version of the emulator, which also demonstrates better performance.

WINE

A development that will certainly exert a significant influence on the propagation of UNIX on PCs both in private and in commercial areas is the Windows Application Binary Interface (WABI). This interface was developed by Sun Microsystems and licensed by the UNIX System Laboratories (USL), so that it will also become available for System V UNIX. WABI supports running MS-Windows programs directly under UNIX. The Windows API (Application Program Interface) calls are converted to corresponding X Window System calls, which has the interesting side effect that these programs can be also used from other X servers due to the network transparency of X11.

WABI

MS-Windows

Early June 1993 the mailing list of the Linux DOS emulator saw the first discussion about porting or full development of a WABI for Linux named WINE. In a few days this project had its own mailing list and the concrete development had begun. A few weeks later a Windows program loader was nearly finished and the first Windows test programs containing a window with a menu were run under Linux.

The developers of WINE will certainly need some time yet before a large MS-Windows application runs. However, simple applications like the Solitaire game and the calculator do run.

Solitaire

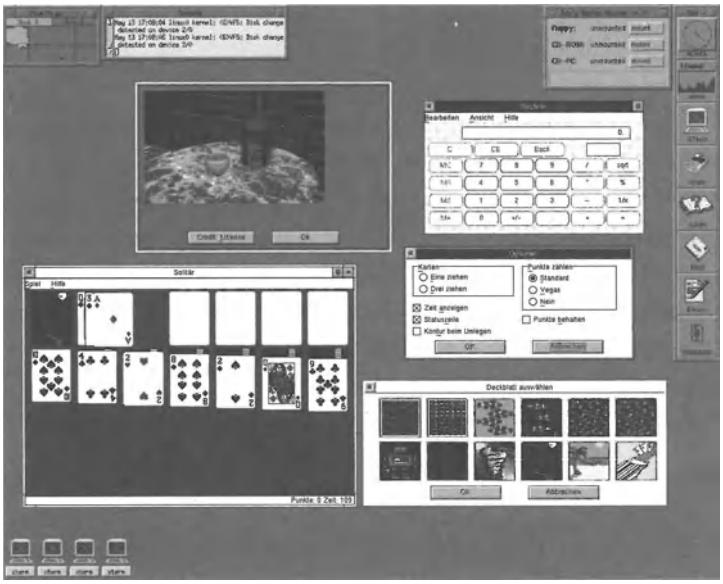


Fig. 4.1 MS Windows Emulator WINE

iBCS2 compatibility

COFF, ELF, XOUT

The development of the iBCS2 (Intel Binary Compatibility Standard) emulator is significantly more advanced than that of WINE. It permits the execution of programs in COFF, ELF or XOUT format.

ELF is the new link format for System V Release 4 programs and libraries, while COFF (Common Object File Format) is supported primarily by version 3.X UNIX systems. This means

run other UNIX programs

that commercial programs for SCO UNIX, XENIX and UnixWare also run under Linux. Examples include WordPerfect in both terminal and X11 versions, Infomix 5.0, and the Motif interface builder XDesigner. Figures 4.2 and 4.3 present some examples.

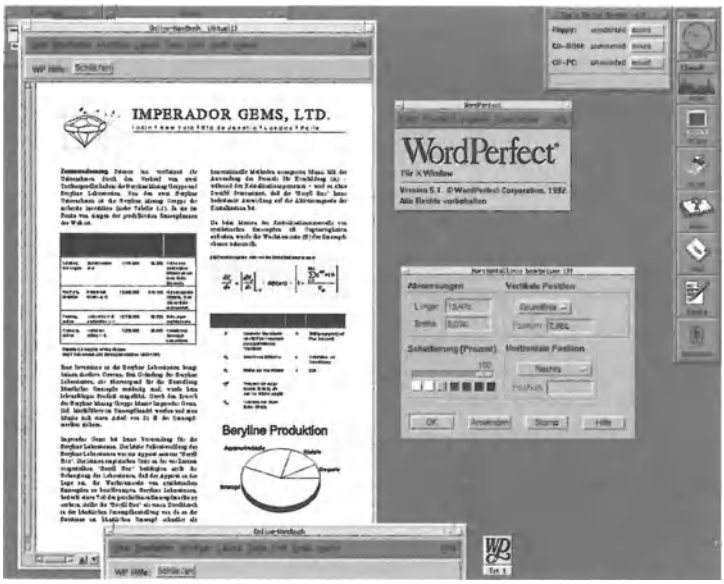


Fig. 4.2 WordPerfect under Linux

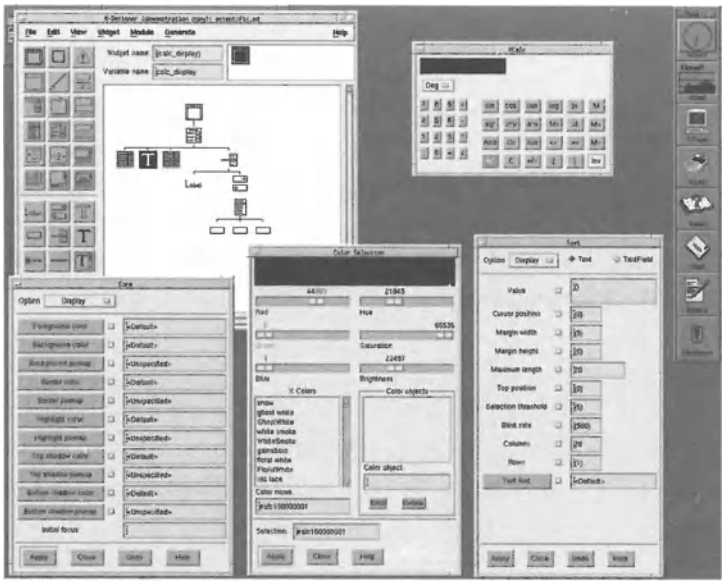


Fig. 4.3 XDesigner SCO binary

Back to the roots

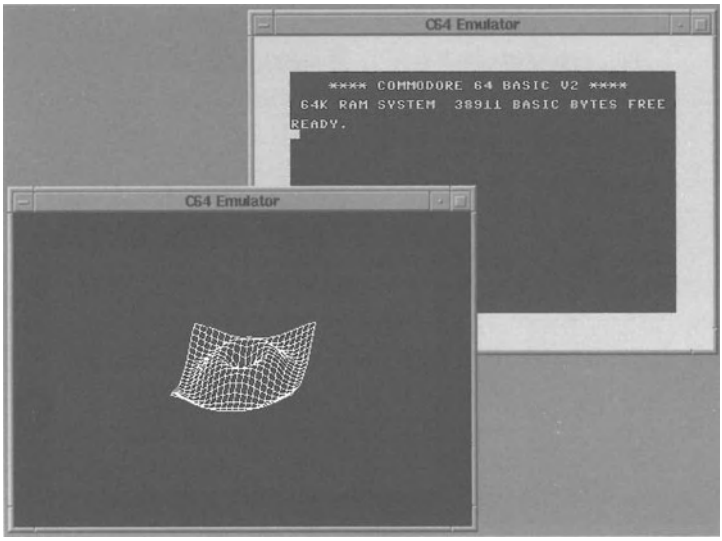


Fig. 4.4 Back to the roots

4.5 Alternative shells

Because their source codes are not freely available and because appreciably more comfortable alternative shells have been developed, Linux seldom uses the original UNIX shells `sh` and `csh`. The following are examples of such alternative shells.

Bash

The bash (Bourne Again shell) evolved, like many other programs, from the GNU project of the Free Software Foundation. It can be considered an extension of the Korn shell, which shares many features of the C shell. Furthermore, the arrow keys allow activation of practical special functions like automatic name and path extension and history scrolling. The following example explains this feature in more detail.

extended Korn shell

To change from the directory /home/stefan to the directory /usr/src/linux, the command cd /usr/src/linux could naturally be entered manually, letter by letter. With automatic path completion in bash it suffices to specify directories only enough to assure that they are unambiguous. Entering <Tab> then automatically extends the path.

automatic path completion

```
/home/stefan> cd /usr/s
```

After entry of <Tab> the above line becomes:

```
/home/stefan> cd /usr/src/
```

Then an l (el) is entered:

```
/home/stefan> cd /usr/src/l
```

After entry of <Tab> the above line becomes:

```
/home/stefan> cd /usr/src/linux/
```

In order to edit the makefile of the Linux kernel in this directory, it suffices to enter only a few characters:

```
/usr/src/linux> emacs M
```

After entry of <Tab> the above line becomes:

```
/usr/src/linux> emacs Makefile
```

Makefile is the only file in this directory that begins with a capital M.

If an attempt to make the extension results in multiple alternatives, there is a warning sound, and, after the <Tab> key is pressed again, bash displays all possible variants.

command history

The up and down arrow keys allow the user to move through the command history. In the above example, pressing the arrow up key once would yield the following on the command line:

```
/usr/src/linux> emacs Makefile
```

Pressing the arrow up again would display the following:

```
/usr/src/linux> cd /usr/src/linux/
```

.bashrc

Environment variables enable the display of the host name and the current path in the prompt. To accomplish this, the following line is added to the file .bashrc in the respective home directory:

```
PS1='\h:$PWD>'
```

The Bourne Again shell offers many other features that cannot be described here in detail. Please refer to the on-line reference manual pages for more information on bash.

TCSH

csh extension

One alternative to the Bourne Again shell is tcsh, an extension of the C shell. Like bash it can automatically extend paths and commands when the <TAB> key is pressed, and the arrow keys allow moving through the command history and editing the command line. Furthermore, various additional functions can be

activated, such as watch mode, where a message is output if a user logs in or out. The display of all alternatives for automatic completion is invoked in `tcsh` with `<Ctrl-D>`, but all keys can be configured. The directory listing of the directory `/usr/src` can be displayed without fully entering the command `cd`:

```
linux1:/home/tul> cd /usr/src/ <Ctrl D>
```

An interesting feature is the automatic correction of erroneously entered commands or file names. Here `tcsh` attempts to use the next closest command or file name. This feature is activated by entering `<Meta-s>`.

error correction

4.6 Extended commands

Many standard UNIX commands that Linux uses emerged from the GNU project and are extensions of the normal commands. For example, under Linux the command `ls` has more than 20 options with which the kind of display, sorting, or handling of symbolic notes can be modified as needed. Many distributions even contain color variants of the `ls` command that displays the lines of the directory listing by color according to the type of the directory element (link, executable file, tar file, subdirectory, etc.).

GNU extensions

Here is an excerpt from the on-line reference manual page of the GNU `ls` command:

```
LS(1L)                                                    LS(1L)

NAME
       ls, dir, vdir, ll, lsf - list contents of directories

SYNOPSIS
       ls  [-abcdgiklmnpqrstuxABCFLNQRSUX1]  [-w  cols] [-T cols]
       [-I pattern] [--all]  [--escape]  [--directory]  [--inode]
       [--kilobytes]  [--numeriC uid-gid]  [--hide-control-chars]
       [--reverse]   [--size]   [--width=cols]   [--tabsize=cols]
       [--almost-all]  [--ignore-backups]  [--classify]  [--file-
       type]   [--ignore=pattern]   [--dereference]   [--literal]
       [--quote-name]                                [--recursive]
       [--sort={none,time,size,extension}]                [--for-
       mat={long,verbose,commas,across,vertical,single-column}]
       [--time={atime,access,use,ctime,status}] [path...]
...
```

GNU tar

automatic compression

The GNU variant of the `tar` command has the option `z` for automatic compression and decompression of tar archives, and the option `M` for establishing a multivolume archive (a single archive that is distributed on multiple storage media).

Gzip

greater efficiency

The command `gzip` replaces `compress` under Linux. This program is compatible with many other compression techniques and has a significantly higher efficiency than `compress`. The command `tar` enables the combination of individual files and subdirectories into a single archive. For a tar archive compressed with the UNIX `compress` command to a size of 1.5 MB, `gzip` usually achieves 900 KB.

All these extended commands are naturally available in source code and could also be compiled on other UNIX machines. The advantage of Linux is that these are used from the start and no additional effort has to be invested in the configuration, compilation, and installation of these utilities.

The list of all enhanced commands and their options would certainly fill a book itself. Hence we refer to the on-line reference manual pages and the individual commands there.

Installation

This chapter describes the most important sources for Linux and the necessary hardware. Then we explain the installation of Linux on the basis of a specific distribution. The next chapter complements this with the configuration of the systems.

5.1 Linux distributions

Because Linux as free software consists of many parts that were developed independently by different people all over the world, there is no "official" Linux distribution that encompasses all the programs available for Linux.

distributions

However, various groups and individuals offer their own installation packages consisting of the Linux kernel along with many utilities, applications, and an installation program. Such a package is called a Linux distribution. The producers of such distributions normally make it available on an ftp server and also ship it on diskettes or CD-ROM for a fee. The Linux distributions residing on ftp servers are normally divided into many subdirectories that each correspond to the contents of an individual diskette.

Almost all of these distributions include at least the kernel and all important utilities and programs that the user needs for an initial installation. In addition, they usually include the GNU C compiler, the graphical user interface X11, and many other programs that can be installed optionally.

installation program

The installation of Linux is normally guided by a menu-driven installation program that permits the selective installation of

subsets of the distribution. Most of these installation programs also offer interactive configuration of the system.

Beyond installation from diskettes, many distributions support the installation of Linux from a CD, from a streamer tape, or via NFS from a remote machine. Depending on the user's experience and how expansive the selected components of the particular installation are, the time required for an installation ranges from one to several hours.

SLS

early distribution

One of the first distributions to make a significant contribution to the propagation of Linux was the SLS (Softlanding Systems) release. For a long time it was the only distribution that enabled a full installation with the C compiler and the graphical user interface. With the appearance of other distributions such as Slackware, the SLS package declined in importance.

source

The current version of SLS always resides in the directory `/pub/linux/packages/SLS` on the ftp server `tsx-11.mit.edu`. Alternatively it can be ordered from SLS as a package of diskettes, a streamer tape, or a CD.

MCC Interim

slim in scope

The University of Manchester assembled the MCC Interim Release, which differs from the SLS distribution and others in that it omits many of the less important programs. For example, the package does not contain X11 or TeX. This makes MCC Interim especially suitable for users who want to install additional programs themselves and want to know exactly what is installed on their hard disk. However, this release can hardly be recommended for novices.

Slackware

A newer distribution that enjoys relative wide propagation is the Slackware package. Originally Slackware was based heavily on

the SLS Release. Meanwhile, however, Slackware uses its own installation routines that give a very good impression. Slackware's color windows, dialog boxes, and selection lists allow the user at the beginning of the installation to specify the components to be installed, the target hard drive and partition, and the country-specific keyboard. The Slackware distribution also supports installation from diskettes, streamer tape, CD-ROM, and NFS.

The distribution is more extensive than that of SLS. In addition to the base system, it offers Tcl/tk, the latest GNU Emacs, and games. Patrick Volkerding assures that the current program versions are contained in the Slackware distribution which certainly presents a challenge due to the rapid rate of development of the Linux system.

Tcl/tk, games, emacs

CD distributions

Many distributions are now available on CD-ROM or as CD subscriptions. The subscriptions deliver a new CD several times a year with the newest version of the Linux kernel and several programs, or even a copy of a whole Linux directory tree from an ftp server. One well-known example is the CD from Yggdrasil Computing, which contains a bootable, installed Linux system as well as the source code of the MIT X11 systems and many GNU utilities.

Yggdrasil

The CD-ROM distributions provide a simple and economical alternative for users who lack an Internet connection. Some Linux CDs sell for under $30, and a CD-ROM drive has become standard equipment on high-end PCs.

The problem that such CD distributions pose, as with diskette distributions that are sold by mail order, is that Linux continues to evolve on a daily basis; in the weeks or months between the assembly of the distribution and production of the CD to the actual shipment, a new version with numerous improvements could already be available on the network.

The Unifix CD distribution is particularly interesting. In addition to the CD, the purchaser receives a small manual and a boot diskette that is used for the first system startup. After the

Unifix

installation of the 5 MB minimum configuration on a partition of the hard disk, the system is ready to use. All application programs can be started directly from the CD. If the computer has enough memory (8-16 MB) then the most important data on the CD are retained in memory, which naturally has a very positive effect on the access time. Changing the CD suffices to make an update. The time-consuming installation of the complete Linux software becomes superfluous.

5.2 Sources

Since Linux's fame has experienced profound growth recently, there are now many possibilities to obtain the above distributions, and additional programs for Linux. Access to an ftp server on the Internet, however, certainly remains the most direct, fastest, and most current approach.

ftp servers

The most important ftp server for Linux in Europe is `nic.funet.fi` in Finland, which was the original Linux server. The newest version of the kernel as well as any alpha releases reside here. MIT's ftp server `tsx-11.mit.edu` contains the most recent GNU C compiler and the Linux C libraries.

A broad palette of Linux software can also be found at `sunsite.unc.edu` at the University of North Carolina. Many software packages that have been ported to Linux are uploaded by their authors/maintainers to this server's incoming directory.

mirrors

These servers are mirrored by several European ftp servers. This means that the data on the ftp servers in Massachusetts and North Carolina are copied to the European servers at regular intervals. This is intended to reduce Internet traffic overall. For European users, a connection within Europe is usually faster and more reliable than direct access to the American servers. The following table lists the most important servers and their mirror servers around the world:

Textual name	Numeric address	Linux directory
tsx-11.mit.edu	18.172.1.2	/pub/linux
sunsite.unc.edu	152.2.22.81	/pub/Linux
ftp.funet.fi	128.214.248.6	/pub/OS/Linux

ftp.cdrom.com	192.153.46.2	/pub/linux
net.tamu.edu	128.194.177.1	/pub/linux
ftp.mcc.ac.uk	130.88.203.12	/pub/linux
src.doc.ic.ac.uk	146.169.2.1	/packages/linux
ftp.informatik.tu-muenchen.de	131.159.0.110	/pub/Linux
ftp.informatik.rwth-aachen.de	137.226.112.172	/pub/Linux
ftp.ibp.fr	132.227.60.2	/pub/linux
kirk.bond.edu.au	131.244.1.1	/pub/OS/Linux
ftp.uu.net	137.39.1.9	/systems/unix/linux
ftp.win.tue.nl	131.155.70.100	/pub/linux
ftp.stack.urc.tue.nl	131.155.2.71	/pub/linux
ftp.denet.dk	129.142.6.74	/pub/OS/Linux
nctucca.edu.tw	140.111.1.10	/Operating-Systems/Linux
nic.switch.ch	130.59.1.40	/mirror/linux
monu1.cc.monash.edu.au	130.194.1.101	/pub/linux
cnuce-arch.cnr.it	131.114.1.10	/pub/Linux
ftp.linux.org	198.182.196.129	/pub

Users without direct access to the Internet who can send and receive e-mail have the option of reaching the Internet ftp servers by way of an ftp mail server. The user simply sends an e-mail to the mail server, which then accesses the ftp server, splits the program into smaller parts, and sends these encoded with the UNIX command `uuencode` as an e-mail to the user.

FTP mail server

To obtain a list of commands that such an ftp mail server understands, the user sends a message with the command `help` in the first line to the ftp mail server. Examples of such ftp mail servers include `ftp-mailer@informatik.tu-muenchen.de` and `ftpmail@decwrl.dec.com`.

Please note, however, that such servers are not suitable for transferring a complete Linux distribution. They are generally limited to transferring small amounts of data.

Mail order

Other sources of Linux and Linux programs are the various commercial dealers who offer Linux diskettes and CDs by mail

order. Their addresses can be found in computer industry periodicals. Some of these dealers offer hard disks and even complete PCs with a running Linux system already installed.

Bulletin Board Service (BBS)

Bulletin board services have sprung up in many larger cities. Some of these services have specialized in Linux or at least offer some Linux programs. Matthias Gmelch has assembled a list that contains an overview of such bulletin board services around the world that offer Linux. This list can be found in the directory `/pub/linux/docs` on the ftp server `tsx-11.mit.edu` and its many mirroring servers as `linux.bbs.list`.

BBS list

5.3 Hardware

One of the most frequently asked question regarding Linux is which hardware Linux supports and requires. The general prerequisite is a PC-compatible computer with an 80386 or newer processor. Linux definitely does not run on old XTs or ATs with 80286 processors because Linux requires task-switching features that have only been available since the 80386.

needs 386 or newer

Memory

The minimal hardware configuration for Linux is an 80386 SX computer with 2 MB of RAM. With such a configuration, the installation can become difficult because 2 MB of RAM is insufficient for many programs. In this case a swap partition or a swap file must be created as soon as possible to be able to start programs and editors needed for the installation.

min. 2 MB of RAM

Normal work in text mode is possible starting at 4 MB of RAM. To be able to work with the X Window System in the normal version, 8 MB of RAM should be available. If the computer has 16 MB of RAM, this delivers a noticeable performance increase under X11.

8 MB or more RAM

Hard disk

By sacrificing the convenience of a graphical user interface in favor of a minimum hard disk installation, the storage requirements can be reduced to about 40 MB. However, for a full installation with X11, the C compiler, and all the tools and utilities, approximately 150 MB needs to be available for Linux. There is hardly a ceiling on what space a diligent network tapper can fill. With access to the Internet it is no problem to fill a 500 MB hard disk with free software. The supply of programming languages, utilities, libraries, and application programs has reached enormous proportions.

150 MB of hard disk

For an economical initial installation, we recommend using a hard disk with an IDE interface. These are available in many versions up to 500 MB and afford adequate performance in single-user operation. This interface was supported from the beginning by the Linux kernel and needs no additional driver. There is even a kernel patch that makes it possible to drive two IDE controllers in parallel as long as their I/O ports and interrupts are configurable.

IDE hard disk

SCSI

For larger and faster hard disks, the Small Computer System Interface (SCSI) is usually used. It permits operating up to seven devices in parallel. These devices can be hard disk drives, streamer tape drives, magneto-optical drives, CD-ROM drives, or scanners.

SCSI operation requires a special SCSI driver in the Linux kernel, but this is contained in all current Linux distributions for the usual host adapters. Exotic host adapters should be avoided because drivers are seldom available. Drivers that are included with the product for DOS or other UNIX versions cannot be used with Linux.

usual SCSI Host adapters

Adaptec controllers 1542b and 1742 pose no problems. There are also drivers for Seagate ST02, Future Domain and others. In case of doubt, refer to the current list of supported hardware that is available on ftp servers under the title "Hardware HOWTO".

Video boards

VGA board with ET4000 chip set

An uncomplicated choice of a video board is to use a simple board with a Tseng ET4000 chipset. Its speed is adequate for normal applications because the X server includes special optimization for this chipset. To date no compatibility problems are known. Many chipsets from other manufacturers are also supported. The current list of supported hardware can be found on ftp servers along with the other HOWTOs under the name "XFree86-HOWTO".

Mach, S3

Video adapters with an accelerator chipset such as S3, Mach8/32/64, or 8514/a deliver significantly better performance. These boards, such as ATI Ultra Pro and ELSA Winner, are particularly fast in the Local Bus or PCI variants. They achieve benchmark values of 100 to 150,000 XStones.

Diamond video boards can be a problem because this manufacturer does not release the necessary specifications, so that no drivers exist for these products.

For an exotic video board or one equipped with little memory, the remaining option is the generic VGA server; however, it supports only 16 colors and a resolution of 640 by 480 pixels, or the mono server, which also supports Hercules graphics.

mouse

Bus mice by all the familiar manufacturers, the usual serial mice, or a PS/2-compatible trackball can be used with Linux.

Bus system (ISA / EISA / PCI)

Linux supports mother boards with the ordinary IDE (ISA), with Local Bus extensions, with the more flexible and faster EISA bus, and with the newer PCI bus.

In the future the new PCI standard will likely replace all currently known bus systems. The PCI bus is processor-independent and significantly faster than the EISA system. It supports data transfer rates of up to 130 MB/s in 32-bit mode.

PCI-boards

Several PCI motherboards have a very fast SCSI chip (NCR 53c810), but it requires a special driver. Although the specifications of the PCI standard were prepared with extreme care, the PCI components available today sometimes deviate from it. This

can cause compatibility problems between the motherboard and peripheral boards. A dedicated PCI HOWTO (see Section 8.4) provides more details on these difficulties as well as advice to potential purchasers of PCI hardware.

Peripheral hardware

streamer

For data backup with streamers, Linux originally afforded only SCSI devices. Meanwhile drivers have been written for more economical floppy streamers, which can be connected to a floppy disk controller.

network boards

The selection of drivers for network boards is quite rich. Besides the frequently-used boards from manufacturers like Novell, 3Com, and SMC, which bought out the production of the old WD boards, several HP and DLink boards are supported. Many no-name boards are compatible with the above and can be used as well.

multi-serial adapter

As is common in the UNIX tradition, Linux also allows the use of ASCII terminals that are connected to the serial port or to a special multi-serial adapter. Drivers are available for these boards as well.

ISDN

One feature that could distinguish Linux from many other systems in the future is its direct kernel support of ISDN boards. Several developers are working on this feature.

sound boards

Even multimedia applications are possible under Linux. The associated peripherals such as sound boards (SoundBlaster, SoundBlaster 16, Adlib, Gravis Ultra Sound or PAS 16) and CD-ROM drives can be operated with the corresponding drivers installed in the kernel. Here, too, it is important to check on the list of supported hardware to determine which CD-ROM drives currently have drivers. SCSI drives are particularly easy to connect.

transputer

Even for exotic hardware such as transputer boards, drivers exist under Linux.

overall configuration

Considering the continuing development in the hardware and software sectors, an 80486 computer with 33 MHz or higher and with 16 MB of RAM currently provides a reasonable and affordable configuration. The hard disk should have at least 200

MB of storage because a permanent shortage of storage precludes productive work. A 14-inch monitor is really too small for a multitasking environment where multiple windows are usually open simultaneously. The Linux X server is able to provide a larger virtual console, and most devices can display 800 by 600 pixels reasonably; thus for a start a 17-inch monitor is not absolutely necessary.

5.4 Installation of the system

This section explains the installation procedure in more detail using the Slackware distribution. With small deviations, this procedure applies to other distributions as well.

installation procedure

The basic steps of a Linux installation are:

- Booting of a minimum Linux system from a boot diskette
- Creation of partitions for Linux
- Creating file systems and swap regions
- Copying the system to the hard disk
- Configuration of the most important system files
- Installation of the boot manager
- Configuration of the graphical user interface
- Creating the users

Boot diskette

The installation procedure begins with the booting of a minimum Linux system that contains only the kernel as well as the most important utilities and the installation program. The Slackware distribution contains a pair of diskettes for this purpose, called the boot and the root diskette. These are provided in 3½-inch or 5¼-inch versions.

boot and root disk

Both the boot diskette and the root diskette are available in various versions. The boot diskettes differ in the drivers that are contained in the kernel; the root diskette, which contains the base file system with the most important auxiliary programs and the

installation script, is available both in a normal version that supports color and in a simpler monochrome version for special cases when the color installation program does not work.

image files

If the distribution is purchased as a diskette package, then the diskettes are configured in a bootable version. Otherwise the installation diskettes contain image files. To create a bootable diskette or one with a Linux file system from such an image file, it must be transferred to the diskette with a special utility as described below. A boot diskette created in this way can no longer be read with normal DOS commands.

The following overview lists all directories with the currently available variants of boot and root diskettes.

```
bootdsks.12
bootdsks.144
rootdsks.12
rootdsks.144
```

The directory `bootdsks.144` contains the following image files for the boot diskette:

```
README
scsi.gz
net.gz
modern.gz
sony535.gz
xt.gz
scsinet.gz
sbpcd.gz
mitsumi.gz
cdu31a.gz
bare.gz
ncr.gz
```

remote installation

For an installation via NFS, the kernel must contain a driver for the network board in the image file for the boot diskette. If the computer has a network board for which the kernel on the boot diskette has no driver, a custom boot diskette can be created. The same applies to a system with a SCSI hard disk and an exotic SCSI host adapter for which none of the boot diskettes have the proper driver. In case of doubt, read the `README` file contained in the same directory as the disk image files, to determine which files are suitable for which hardware configuration.

root disks

There are also different files for the root diskette, although here the distinction is only between a diskette with color support

and a monochrome diskette. In most cases the file for the color installation program can be used: for a 3½-inch drive the file `rootdsks.144/color144.gz`. The color version requires more storage space, so this diskette contains fewer auxiliary programs, which is usually no problem.

boot disks

The other directories under `kernels` contain files for creating special boot diskettes or additional tools and information. The file `kernels/README.NOW`, for example, contains a description of how the user can create a custom boot diskette with an adapted kernel.

If the user has selected the correct files for the boot and root diskettes, then they have to be transferred to diskettes either under DOS with `rawrite.exe` or under UNIX with the `dd` command. The diskettes to which the files are transferred must first have been formatted under DOS. The source files must be decompressed with `gzip`. The following example demonstrates this procedure.

```
stef1:bootdsks.144>ls
README      cd.gz       net.gz      scsinet.gz
bare.gz     cdscsi.gz   scsi.gz     xt.gz
stef1:bootdsks.144>gzip -d bare.gz
stef1:bootdsks.144>ls
README      cd.gz       net.gz      scsinet.gz
bare        cdscsi.gz   scsi.gz     xt.gz
stef1:bootdsks.144>dd if=bare of=/dev/fd0 bs=8192
180+0 records in
180+0 records out
stef1:bootdsks.144>
```

reboot

LILO

Once the diskettes have been created, the diskette can be placed in the boot drive and the computer can be rebooted. The first message to be displayed comes from LILO, the Linux boot manager and loader installed on the boot diskette. The following example shows the first message during booting.

```
LILO

Welcome to the Slackware Linux 2.0.0 Bootkernel disk!

If you have any extra parameters to pass to the kernel, enter them at the
prompt below. For instance, you might need something like this to detect the
hard drives on PS/1 and ValuePoint models from IBM:

  ramdisk hd=cyl,hds,secs   (Where "cyl", "hds", and "secs" are the number of
                             cylinders, sectors, and heads on the drive.)

Also, in a pinch, you can boot your system with a command like:
  mount root=/dev/hda1

If you would rather load the root/install disk from your second floppy drive:
  drive2

DON'T SWITCH ANY DISKS YET! This prompt is just for entering extra parameters.
If you don't need to enter any parameters, hit ENTER to continue.

boot:
```

kernel parameters

The Linux Loader now waits for input. For a normal installation in which the first disk drive is also to be used for the root diskette, it suffices to press <Return>. In some cases, for example, if a hard disk or a network board is not correctly recognized, special parameters have to be passed to the kernel. These parameters are listed after the specification of the boot option. An entry after the boot prompt has the following structure:

```
BootSelectParameter=Value,Value, -Parameter=Value,Value,...
```

An example was shown in the above Linux Loader message. The following is another example:

```
ramdisk ether=5,0x320
```

This tells the driver for the Ethernet board in the kernel that the board is using interrupt 5 and IO address 0x320.

load kernel

If the choices are correct and any necessary options have been passed, then the Linux Loader loads the kernel. The kernel first decompresses itself and then initializes its individual components, displaying corresponding messages all the while. The messages of the device drivers indicate which devices were recognized and successfully initialized. The sequence of displays during the boot process could look like this:

```
Console: colour EGA+ 80x25, 8 virtual consoles
Serial driver version 3.99a with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16450
lp_init: lp1 exists (0), using polling driver
PS/2 auxiliary pointing device detected -- driver installed.
Calibrating delay loop.. ok - 33.22 BogoMips
Memory: 7140k/8192k available (496k kernel code, 384k reserved, 172k data)
This processor honours the WP bit even when in supervisor mode. Good.
Floppy drive(s): fd0 is 1.44M
floppy: FDC version 0x90
Swansea University Computer Society Net2Debugged [1.30]
IP Protocols: ICMP, UDP, TCP
eth0: SMC Ultra at 0x340, 00 00 C0 13 FD 6D, IRQ 5 memory 0xe0000-0xe3fff.
smc-ultra.c:v0.07 3/1/94 Donald Becker (becker@super.org)
Checking 386/387 coupling... Ok, fpu using exception 16 error reporting.
Linux version 1.0 (root@kebap) #2 Thu Apr 14 23:17:33 GMT+0200 1994
Partition check:
  hda: hda1 hda2 hda3 hda4
VFS: Mounted root (ext2 filesystem) readonly.
Adding Swap: 7344k swap-space
```

file system

After the kernel has been successfully loaded and started, a message appears on the screen that prompts the user to replace the boot diskette with a root diskette containing the basic file system.

```
Please remove the boot kernel disk from your floppy drive, insert a
root/install disk (such as one of the Slackware color144, colorlite,
tty144, or tty12 disks) or some other disk you wish to load into a
ramdisk and boot, and then press ENTER to continue.
```

When the root diskette is ready, its contents are read into a ramdisk, thus freeing the floppy disk drive for other diskettes.

```
RAMDISK: 1474560 bytes, starting at 0x1d0000
RAMDISK: Loading 1440 blocks into Ram Disk
```

With the completion of booting, a welcome message appears on the screen that explains subsequent steps.

```
Welcome to the Slackware Linux installation disk, (v. 2.0.0)

###### IMPORTANT! READ THE INFORMATION BELOW CAREFULLY. ######
- You will need one or more partitions of type "Linux native" prepared. It is
  also recommended that you create a swap partition (type "Linux swap") prior
  to installation. Most users can use the Linux "fdisk" utility to create and
  tag the types of all these partitions. OS/2 Boot Manager users, however,
  should create their root Linux partition with OS/2 "fdisk", add it to the Boot
  Manager menu, and then use the Linux "fdisk" to tag it as type "Linux native".
  OS/2 users may make their non-root Linux partitions with Linux "fdisk".
- If you have 4 megabytes or less of RAM, you MUST activate a swap partition
  before running setup. After making the partition with fdisk, use:
mkswap /dev/<partition> <number of blocks> ; swapon /dev/<partition>
- Once you have prepared the disk partitions for Linux, and activated a swap
  partition if you need one, type "setup" to begin the installation process.
- If you want the install program to use monochrome displays, type:
TERM=vt100
  before you start "setup".

You may now login as "root".

slackware login:
```

To continue, the user enters the user name `root`. This brings another message :

login

```
Linux 1.1.19. (Posix).

If you're upgrading an existing Slackware system, you might want to
remove old packages before you run 'setup' to install the new ones. If
you don't, your system will still work but there might be some old files
left laying around on your drive.

Just mount your Linux partitions under /mnt and type 'pkgtool'. If you
don't know how to mount your partitions, type 'pkgtool' and it will tell
you how it's done.

To start the main installation, type 'setup'.

#
```

All subsequent steps take place under Linux.

Partitioning

Installing Linux requires at least one free partition. However, it is a good idea to create at least three partitions: one for the root file system, one for the home file system, and one as a swap partition. Then a new installation of the system can leave the /home file system unscathed and ready for remounting after the installation. This also retains all user files.

three partitions

If the hard disk does not contain any partitions that can be used for Linux, then the partitions must be created first with the Linux command `fdisk`. The example below demonstrates the creation of partitions for Linux. We assume that one partition for DOS exists on the hard disk.

fdisk

The partition type specified for a new partition is 83 (Linux native). The intended swap partition is changed to type 82 (Linux swap). A partition's type is read by the installation program, which thus recognizes which partitions can be used for the installation.

If partitions for Linux already exist, for example, because they were created under DOS with `fdisk`, then the type of any partition to be used for a Linux file system should be set to 83 with the fdisk command under Linux.

The following terminal capture reflects the execution of `fdisk` under Linux:

```
# fdisk
Using /dev/hda as default device!

Command (m for help): p

Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1           1       1     418  102392+   6  DOS 16-bit >=32M

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 2
First cylinder (419-978): 419
Last cylinder or +size or +sizeM or +sizeK (419-978): +100M

Command (m for help): p

Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1           1       1     418  102392+   6  DOS 16-bit >=32M
/dev/hda2         419     419     836  102410   83  Linux native

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 3
First cylinder (837-978): 837
Last cylinder or +size or +sizeM or +sizeK (837-978): +8M

Command (m for help): p

Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1           1       1     418  102392+   6  DOS 16-bit >=32M
/dev/hda2         419     419     836  102410   83  Linux native
/dev/hda3         837     837     870    8330   83  Linux native

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 4
First cylinder (871-978): 871
Last cylinder or +size or +sizeM or +sizeK (871-978): 978
```

```
Command (m for help): p

Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes

   Device Boot  Begin    Start     End  Blocks   Id  System
/dev/hda1           1        1     418  102392+   6  DOS 16-bit >=32M
/dev/hda2         419      419     836  102410   83  Linux native
/dev/hda3         837      837     870    8330   83  Linux native
/dev/hda4         871      871     978   26460   83  Linux native

Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): L

 0  Empty            8  AIX             75  PC/IX           b8  BSDI swap
 1  DOS 12-bit FAT   9  AIX bootable     80  Old MINIX       c7  Syrinx
 2  XENIX root       a  OPUS            81  Linux/MINIX     db  CP/M
 3  XENIX usr       40  Venix 80286      82  Linux swap      e1  DOS access
 4  DOS 16-bit <32M 51  Novell?         83  Linux native    e3  DOS R/O
 5  Extended        52  Microport       93  Amoeba          f2  DOS secondary
 6  DOS 16-bit >=32 63  GNU HURD        94  Amoeba BBT      ff  BBT
 7  OS/2 HPFS       64  Novell          b7  BSDI fs
Hex code (type L to list codes): 82
Changed system type of partition 3 to 82 (Linux swap)

Command (m for help): p

Disk /dev/hda: 14 heads, 35 sectors, 978 cylinders
Units = cylinders of 490 * 512 bytes

   Device Boot  Begin    Start     End  Blocks   Id  System
/dev/hda1           1        1     418  102392+   6  DOS 16-bit >=32M
/dev/hda2         419      419     836  102410   83  Linux native
/dev/hda3         837      837     870    8330   82  Linux swap
/dev/hda4         871      871     978   26460   83  Linux native

Command (m for help): w
The partition table has been altered!

Calling BLKRRPART ioctl() to re-read partition table
Syncing disks
Reboot your system to ensure partition table is updated
#
```

All partitions, including the logical drives of an extended partition, are simply numbered sequentially under Linux and can be used like a primary partition. All partitions are mapped onto files in the directory `/dev`, their names beginning with `hd` for normal hard disks or `sd` for SCSI hard disks. More detailed information can be found in Section 2.5.

Creating file systems

Before a partition is capable of storing files, a file system must be created on it. In general this is done by selecting the appropiate menu item in the installation script.

Alternatively, a file system can be created manually with the command `mkfs`. This command is only a front end that invokes the respective program to create the selected file system according to the file system type parameter (Option `-t`) that is

mkfs

passed to it. For an Extended-2 file system, for example, this is the command `mke2fs`.

```
# mkfs -t ext2 -c /dev/hda4
```

The parameter -c activates the verification of the blocks on the hard disk that are used for the file system.

Creating the swap partition

mkswap

Similarly, a swap partition is created with the command `mkswap`. In addition to the device, the number of blocks in the partition must be specified. This can be obtained easily by invoking the program `fdisk` with the option `-s`, which outputs the size of a partition in blocks.

```
# fdisk -s /dev/hda3
8330
# mkswap /dev/hda3 8330
Setting up swapspace, size = 8523776 bytes
#
```

swapon

To activate the new swap partition, the user enters the command `swapon` followed by a parameter. The current state of main memory is displayed with the command `free`.

```
# swapon /dev/hda3
# free
             total       used       free     shared    buffers Mem:
              7060       5248       1812        888       2300
Swap:         8324          0       8324
#
```

8 MB or more

For machines with 8 MB or more of RAM available, both the swap partition and the file system could be prepared and activated in the installation program. If the menu item ADDSWAP is selected, the hard disk is searched for a partition of type *Linux swap*. If such a partition is found, the user can register it as the swap partition. Then this partition can be initialized with `mkswap` and activated with `swapon`.

4 MB or less

If the machine only has 4 MB or less of RAM, then a swap partition has to be created and activated before the invocation of

the installation program. The available memory will not suffice otherwise.

Copying to the hard disk

Once the swap partition and the file system are set up, the Linux system can be copied onto the hard disk by invoking the installation program via the command `setup`.

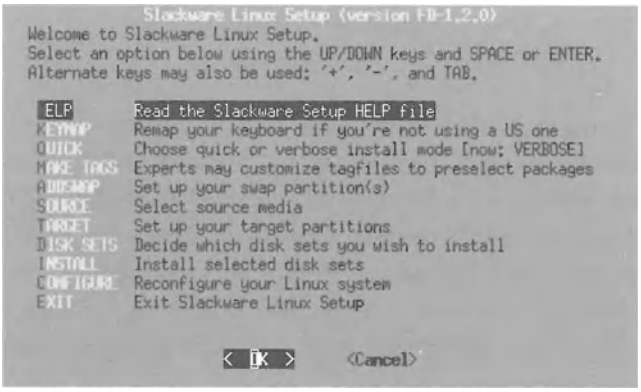


Fig. 5.1 Main menu of the installation program

The most important items in the main menu are the selection of the source medium, the selection of the target partition, the choice of the components to be installed, and the start of the actual installation. These menu items are automatically linked; that is, if the first point is selected, the installation program automatically moves to the next item when the first item is complete.

The third menu option allows the selection of a quick or a verbose installation mode. We recommend leaving the installation mode at verbose.

installation mode

Fig. 5.2 Installation mode (quick/verbose)

NFS, CD-ROM, streamer tape

The source medium could be the hard disk if the files from the diskettes were copied there beforehand, the diskettes themselves, another computer on the network mounted with NFS, a CD-ROM, or a streamer tape drive. Installation from a streamer tape is somewhat more tedious than the other possibilities and is explained in a separate `README` file located in the directory `install/tape`. The special version of the root diskette needed for a streamer tape installation is also located in this directory under the name of `tape144`.

Fig. 5.3 Source media selection

network installation

An installation via NFS requires the entry of several network parameters such as the IP address of the target computer and that of the NFS server, the network mask, the broadcast and network address, and the path of the distribution on the NFS server. These

terms are explained in Chapter 3. In case of doubt, call on the network administrator who knows the local setup for advice.

The selection of the target partition is menu-driven. The installation program independently searches the system for all partitions of type *Linux native* and displays these for selection:

target partition

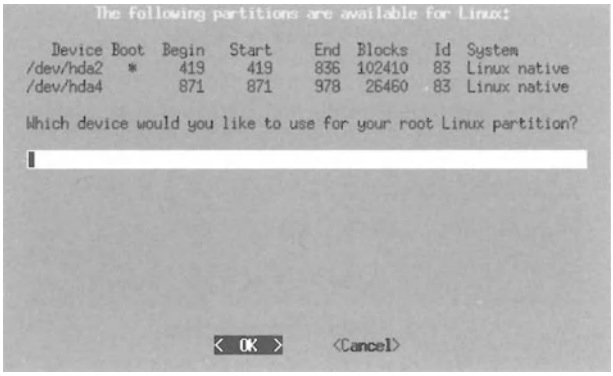


Fig. 5.4 Selection of root partition

Likewise the packages to be installed are selected from a list of all possible packages of the distribution accompanied by short descriptions.

package selection

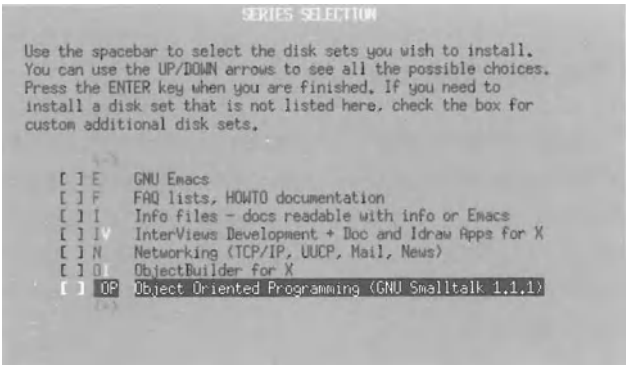


Fig. 5.5 Selection of disk series

Next the menu item *Install* is invoked. The user is presented with a selection of modes in which the copying procedure can be carried out. For an initial installation we recommend the *Normal*

mode as the best choice. This mode installs the fundamental system components automatically and asks the user about optional elements. Before each copying step the installation program displays information about the subpackage being installed, which eases deciding about optional components.

Once the selected packages are installed, the configuration of the system begins. Here various parameters and links are set, such as for the connection of a modem or a mouse.

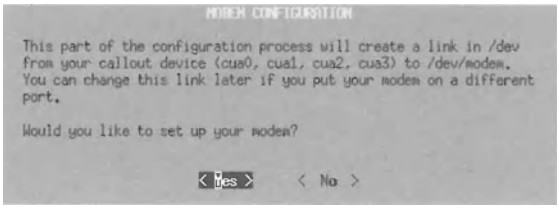


Fig. 5.6 Modem configuration

LILO

The user also has the option of installing the Linux Loader via a menu (see also the installation of the Linux Loader below).

With the completion of the installation, the program automatically returns to the main menu, where the user has the opportunity to quit. This ends the installation of the Slackware distribution and the computer can be rebooted.

5.6. The boot manager (LILO)

Linux Loader

The Linux Loader (LILO) permits Linux to be loaded immediately on booting. If there are multiple operating systems on the hard disk, then LILO can manage the selection of the system to be started. Besides Linux, LILO can load DOS, OS/2, or a different PC-UNIX variant, and can even boot these from a second hard disk.

select boot system

The functional principle is similar to that of the OS/2 boot managers. Instead of loading an operating system immediately, LILO is started first, which then offers a selection of all registered operating systems and configurations.

Operation

The Loader first presents itself with the word "LILO" across the screen. Then the user has a predefined amount of time to press one of the keys <Shift>, <Alt>, <Ctrl>, or <AltGr> not to boot the standard configuration, which lets the user select another partition or configuration. Then the Loader prompts the user with the configurable message "boot:" to enter a boot variant.

As soon as the boot prompt appears, pressing the tabulator key displays a list of available alternatives.

boot prompt

```
LILO boot:
linux       linux-old       dos
boot: linux
Loading linux
```

If one of the above keys is not pressed during the predefined time, then the first defined operating system is loaded.

Installation

LILO is normally installed in the directory /sbin: The configuration file is /etc/lilo.conf.

We will now describe a typical installation for a system with two IDE hard disk drives with Linux installed on the second drive.

First we modify the file /etc/lilo.conf. The following is an example of a configuration file for the Linux Loader:

```
boot = /dev/hda
delay = 100
compact
image = /vmlinuz
        label = linux
        root = /dev/hdb1
        vga=1
image = /vmlinuz.old
        label = linux-old
        root = /dev/hdb1
        vga=1
other = /dev/hda1
        label = DOS
```

The first line specifies where LILO will be installed. The choices are at the start of a hard disk drive and thus in the master

installation in MBR

boot record (MBR), or at the start of a partition. A simple solution is to install LILO directly in the MBR of the first hard disk drive. This means that LILO will be started first, regardless of which partition in the partition table has been designated as the active one.

Caution! Installing LILO in the master boot record overwrites the original DOS MBR. However, this MBR can be restored with fdisk under DOS (see page 86).

If LILO is used alongside another boot manager such as that of OS/2, installing Linux in the MBR does not work (see page 86). In this case LILO is installed at the start of a partition.

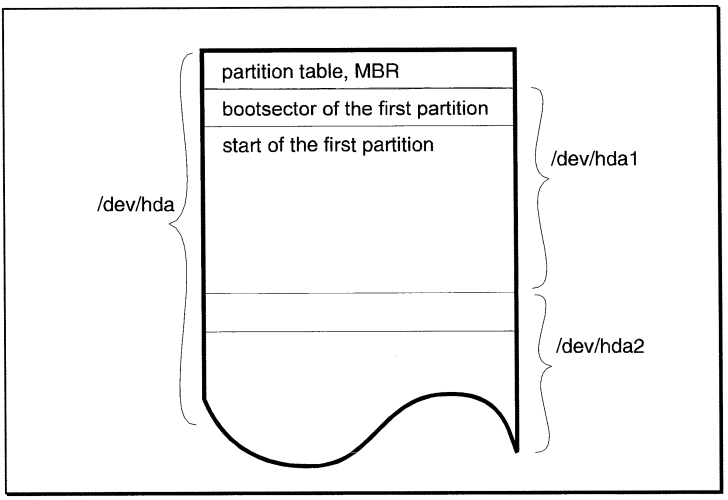


Fig. 5.7 Example of the structure of the first hard disk

delay and compact

`delay` defines the time in tenths of seconds that LILO is to wait to determine whether one of the above keys has been pressed, before automatically reverting to the first boot alternative. `compact` is an optimization that prevents every sector from being read individually.

The entry `image = /vmlinuz` and the subsequent indented lines indicate the first boot alternative. The kernel image file to be booted is called `vmlinuz` and resides in the root directory. The boot alternative designation that needs to be entered on booting is "linux".

The second boot alternative is the file `/vmlinuz.old`. This is an older kernel image file that serves as a reserve for emergencies in case the current kernel fails to work.

The third alternative is booting from the DOS partition of the first IDE hard disk. The label "other" causes the loader of some other operating system to be started rather than a Linux kernel.

DOS

Kernel image files

A kernel image is a file that contains the actual operating system kernel along with an initialization and loading program. This file is created when the kernel is compiled whose source code resides in `/usr/src/linux` (see "Configuration" and "Compilation" in Section 6.2).

The cautious user should also enter the old version of the kernel in the LILO configuration file. Then, in case there was an error in the compilation or configuration and the system refuses to boot with the new kernel, the user can still boot with the old kernel and correct the error.

backup kernel

This backup approach is supported by the makefile of the kernel. On invocation of `make zlilo` in the kernel source code directory `/usr/src/linux`, after compilation the new kernel image file is copied to `/vmlinuz` and the old one is renamed from `/vmlinuz` to `/vmlinuz.old`.

new kernel

Activating the Loader

To activate the entries in the configuration files, LILO must be reinstalled. Invoke the command `lilo`, which writes the actual Loader with its current configuration to a boot sector or the MBR.

```
dirk1:# /sbin/lilo
Added linux
Added linux-old
Added DOS
dirk1:#
```

Removing LILO

restoring MBR

If LILO was installed in the MBR and should now be removed for whatever reason, then the previous contents of the MBR must be restored. If DOS is available on another hard disk or partition, then the simplest solution is to invoke the DOS fdisk program with the option /mbr. This installs a new master boot record and overwrites LILO.

Alternative boot manager

OS/2 boot manager

To install a boot manager other than LILO, such as the OS/2 boot manager, LILO must not be installed in the MBR; otherwise it will definitely be started as the first boot alternative. Instead, LILO can be installed at the beginning of the Linux root partition.

The OS/2 version of fdisk should be used to set up the partitions. Afterwards the OS/2 boot manager can be initialized and the boot partition activated. Then the Linux partition, where the Linux Loader was installed, should be registered as well. On booting the OS/2 boot manager is activated; from here the user can either select OS/2 or, via LILO, start the Linux system.

Check the LILO user's guide or the README File included with the LILO distribution for more information.

Configuration

Once the system files have been copied onto the hard disk and the Linux Loader has been installed, some of the configuration files need to be adapted. This fine-tunes the operating system to the available hardware.

6.1 General configuration

configuration files

Most configuration files belong in the directory `/etc`. Many of these files are fixed in content and normally do not need to be changed. The appendix contains a brief description of these files. This chapter discusses the configuration modifications that have to be carried out after the installation of Linux on normal PC hardware.

File systems

mounting

When the system starts up, one of the `rc` scripts (see Chapter 7) invokes the command `mount -a`, which mounts all file systems specified in the file `/etc/fstab` into the directory tree. This file enumerates all available file systems along with their options, which are passed as parameters to the `mount` command.

multiple file systems

If multiple file systems have been created as described in the previous chapter, these should be entered in the file `/etc/fstab`. The following example (excerpt from the file /etc/fstab) shows these entries for a computer with two file systems:

```
/dev/sda2       /               ext2    defaults
/dev/sda5       /home           ext2    defaults
none            /proc           proc    defaults
```

The entry that mounts the `/proc` file system is important because several commands, such as a variant of `ps`, are dependent on this directory. System information is represented here in the form of files and subdirectories.

If the parameter `noauto` is used as an option in the file `/etc/fstab`, the specified file system is not mounted automatically with `mount -a` but has to be mounted explicitly. This is used for removable media or special NFS mounts. For a complete description of the options, see the manual page for the `mount` command.

home directory

The `/home` directory, where the files for individual users are normally stored, should be located on a separate file system. If the need should arise, the system can be re-installed without a loss of important user data.

Swap space

limited memory

If the target computer has 8 MB or less of RAM, it is necessary to create a swap partition or a swap file (if this was not already done during installation). Otherwise the machine will not have enough memory to execute multiple programs simultaneously under the graphical user interface. Four MB of memory does not even suffice to recompile the kernel while running an editor at the same time. Even with 16 MB it is a good idea to extend virtual memory with a swap partition.

swapon

The command `swapon` activates the swap space (see also "Creating a swap partition" in Section 5.4). Entering the swap partition in the file `/etc/fstab` like a file system effects its automatic activation at system startup.

The following serves as an example of an `/etc/fstab` file with entries for ordinary file systems as well as a swap partition and NFS mounts:

```
/dev/sda2              /          ext2       defaults
/dev/sda5              /home      ext2       defaults
none                   /proc      proc       defaults

/dev/sda3              none       swap       defaults

sun1:/home             /sun       nfs        defaults
linux1:/               /linux1    nfs        defaults
```

fstab file

The entries largely correspond to the parameters of the `mount` command. The first column specifies the device to be mounted. For an NFS mount the first column designates the host, followed by a colon and the respective directory. The second column gives the path specification for where in the file system it is to appear. The third column identifies the type of the file system. Most file systems permit the specification of additional options in the last column. If no special parameters are desired, then this column should contain the word `defaults`, which activates the standard defaults.

Login

login options

If the Shadow Password package included in many distributions has been installed, the many options concerning user login are set in the file `login.defs` in the `/etc` directory. For example, this file specifies how long the login prompt is disabled (the delay) after the entry of an incorrect password and the devices from which the user `root` can log in. Notes can be found in the file `/etc/login.defs` itself or in the corresponding on-line reference manual page. The following is a small excerpt from this file.

```
# Delay in seconds for which the login prompt
# is blocked after an incorrect password.
FAIL_DELAY   2

# Devices from which a login as root is permissible
#CONSOLE     /etc/consoles
#CONSOLE     console:tty01:tty02:tty03:tty04
CONSOLE      tty1:tty2:tty3:tty4:tty5:tty6:tty8
      .
#  Files that are output after login
MOTD_FILE    /etc/motd
```

The alternative login package, which does not support shadow passwords, does not include this file. The corresponding parameters were determined at compilation.

Keyboard layout configuration

run-time configuration

Since Version 0.99.10 of the Linux kernel, keyboard settings are no longer bound to the compilation of the kernel, but can be changed at run time. The command `loadkeys` serves this purpose by loading a table containing a keyboard layout. On system startup, this command should be invoked from one of the `rc` files. For example, German-speaking users would use the file `gr-latin1.map` as their keyboard layout; it provides a normal German keyboard with umlauts as defined by ISO Latin-1.

The Slackware package contains the `loadkeys` command in the directory `/usr/bin/loadkeys` and the map files in the directory `/usr/lib/kbd/keytables`. To prevent having to reload the keyboard tables on every restart, an appropriate command should be entered in the `/etc/rc.d/rc.local` or `/etc/rc.d/rc.keymap` file. The following example demonstrates loading the German keyboard layout:

```
# Load country-specific keyboard table
/usr/bin/loadkeys /usr/lib/kbd/keytables/gr-latin1.map
```

6.2 The kernel

Like the commands, utilities and all other Linux programs, the kernel, the actual heart of Linux, is available as source code for free. The kernel was written primarily in C, with some small parts in Assembler. In addition to the scheduler, which manages the switching between running processes, the kernel contains the drivers for peripheral devices and the routines for managing the file systems.

Configuration of the kernel

Another step after the base installation of the operating system is the configuration and compilation of the kernel. However, in many cases this can be omitted. In order to achieve optimal fine-tuning to the available hardware, compilation is absolutely recommended. This allows the administrator to dispense with drivers that are not needed, to add new drivers, and to modify settings of drivers. This customizing reduces the memory requirements of the kernel and accelerates booting.

finetuning

Normally, the source code of the kernel is located in the directory `/usr/src/linux`. This directory also contains a configuration script that is invoked by the makefile and significantly simplifies the customizing of the kernel. This script is activated with `make config` and permits the setting of the following options and more:

source code

- Supported file systems
- TCP/IP support
- Use of coprocessor emulator
- Optimization for 80486 processors
- SCSI support
- Drivers for SCSI and network boards
- Parameters for specialized interface cards
- Settings for sound boards

options

Compilation

Once all necessary specifications have been made, the dependencies among the individual part of the source code have to be determined anew. This process is started with the invocation of `make dep` and requires being in the main directory of the kernel source code (`/usr/src/linux`).

Since old object files could still be present when the configuration is modified, these should first be removed with `make clean`. Then the actual compilation process can start. Here the makefile plays a central role, as during installation and compilation of most other programs: It contains the dependencies

removing old files

between the individual source code files and helps to coordinate various scripts for configuration and installation of the kernel.

By default a compressed kernel is generated. An integrated routine decompresses the kernel during booting. This feature of the Linux kernel allows a minimal but complete Linux system that supports all hardware (network, streamer, SCSI devices, CD-ROM) to fit onto a single boot disk. The time lost in decompression is insignificant.

make options

The following are the most important variants in the invocation of the `make` command.

- **`make dep`** - Determines anew the dependencies among source code files, which should be done after any modification in the configuration of the kernel.
- **`make clean`** - Deletes all object files. On the next compilation, all source code files are compiled from scratch.
- **`make`** - Compiles the kernel and creates a new kernel image in the directory `/usr/src/linux` with the name `zImage`. This procedure takes place with many other invocations that later use this image file.
- **`make zlilo`** - Creates a new kernel image as above and copies it to `/vmlinuz`. If an old file of the same name is present, it is moved to `/vmlinuz.old`. Then the installation script of the Linux Loader is invoked, so that with the next system startup the new kernel can be booted.
- **`make disk`** - On completion of the compilation, the image file is written directly to the disk currently in drive A. Before this command is invoked, an empty, formatted diskette should be placed in drive A.

6.3 Daemons

The settings of some daemons that are started with the booting of the system are stored in configuration files. However, since no terminal is assigned to the daemons, they normally do not output error messages, but send all error messages and other reports to the syslog daemon. To simplify the detection of errors in the

daemon output

configuration of daemons and other programs, the syslog daemon should be configured next.

Syslog daemon

messages from other daemons

The syslog daemon `syslogd` can write a message that it receives from another daemon to a file, send it by e-mail to certain users, or display it directly on the console. Individual settings can be made to determine where messages are output for each unit that sends syslog a message and for the priority of such a message. These settings are specified in the file `/etc/syslog.conf`.

A simple configuration that usually suffices is to collect all messages of a certain priority class in one file. The entries necessary in `/etc/syslog.conf` to achieve a base installation of the syslog daemon would look like this:

```
*.alert          /var/log/alert
*.emerg          /var/log/emerg
*.crit           /var/log/crit
*.err            /var/log/err
*.warning        /var/log/warning
*.notice         /var/log/notice
*.info           /var/log/info
*.debug          /var/log/debug
```

units, priorities

All entries consist of the specification of a unit, a priority, and the target of the messages. An overview of all defined units and priorities can be found in the on-line reference manual page for `syslogd` or for the file `syslog.conf`.

debug mode

The files in the directory `/var/log` must exist when the syslog daemon starts. To create an empty file, it is easiest to use the command `touch` with the respective file as its parameter. To check whether the settings in the configuration file are correct, terminate the syslog daemon with the `kill` command and then restart it with the option `-d`. This starts the daemon in debug mode; syslog then displays a matrix showing which messages are written to which files or sent to which users.

log files

Since in the above configuration all messages are appended to files, the administrator should check that these files do not grow out of proportion. We suggest using a script that `crond` executes regularly to move the log files to another directory and then to

create them anew in `/var/log`. Afterwards the syslog daemon must be notified by means of `/etc/syslogd.reload` to reopen its message files. If problems arise later, the old log files can still be referenced.

Printer daemon

The general definition and configuration of printers takes place in the file `/etc/printcap`. This file specifies, for example, whether to include a cover sheet with each new print job or whether a form feed should follow a print job. Furthermore, multiple printer queues can be created, each with its own filter program. In order to enable a computer to access print servers,

network printer

which provide printers via the network, the client machine must be entered in the file `/etc/hosts.lpd` of the print server. This file contains the names of all computers that have access to the printer.

The following example shows the file `/etc/printcap` of a Linux machine that does not possess its own printer, but has

shared printer

access to a printer on a workstation.

```
ibmdr|risc1lpr|ibm 4019::lp=:rm=risc1:\
        sd=/usr/spool/ibmdr:lf=/usr/spool/ibmdr/ibmdr-log:
ibmps|risc1ps|ibm 4019 PostScript::lp=:rm=risc1:\
        rp=ps:sd=/usr/spool/ibmps:lf=/usr/spool/ibmps/ibmps-log:
```

The on-line reference manual page for `printcap` contains a list of options that can be specified in this file.

international character sets

International character sets (e.g., French, Spanish, German) pose a problem in outputting text files to a printer. For German, this includes the umlauts. The installation of a corresponding filter delivers satisfactory results here. Such a filter can be realized in various ways. The following example for German lists a simple C program that converts umlauts and sends the printer a carriage return (CR) after each line. Alternatively, the UNIX command `tr` for character conversion could serve this purpose.

```
/*************************************************************
 *
 * Umlaut conversion for EPSON printer
 *
 *************************************************************/

#include <stdio.h>

main (int argc, char *argv[])
{
  int ch;

  while ((ch = getchar ()) != EOF)
  {
    /* printer needs CR+LF */
    if (ch == '\n')
      putchar ('\r');

    /* convert ISO to PC */
    switch (ch)
    {
      case 228: /* "a */
        ch = 132;
        break;
      case 246: /* "o */
        ch = 148;
        break;
      case 252: /* "u */
        ch = 129;
        break;
      case 196: /* "A */
        ch = 142;
        break;
      case 214: /* "O */
        ch = 153;
        break;
      case 220: /* "U */
        ch = 154;
        break;
      case 223: /* sz */
        ch = 225;
        break;
      case 167: /* paragraph */
        ch = 21;
        break;
      default:
        break;
    }
    putchar (ch);
  }
}
```

if and of

Such a filter is linked via the option `if` or `of` in the `/etc/printcap` file. An `of` filter is initialized only once; an `if` filter is restarted for each print job.

The following figure shows the linking of a filter in `/etc/printcap`.

```
#
#       /etc/printcap
#
lp:lp=/dev/lp1:sf:sd=/usr/spool/lp:mx=0:sh

# Text queue
txt:lp=/dev/lp1:sf:sd=/usr/spool/txt:\
      if=/usr/spool/lp/epson:mx=0:sh

# PostScript queue
ps:lp=/dev/lp1:sf:sd=/usr/spool/ps:\
     if=/usr/spool/lp/PostScript:mx=0:sh
```

Registering multiple printer queues enables switching between filters as needed. The choice of the correct queue is specified in a parameter of the `lpr` command. The following allows printing a text with umlauts:

```
linux1:/home/tul> lpr -Ptxt Umlaut.txt
```

PostScript

With the right filter, an ordinary matrix, ink jet, or laser printer can easily be converted to a full-scale PostScript printer. The PostScript interpreter Ghostscript is registered as the filter, which cannot be done directly, but via a shell script.

```
#!/bin/sh
exec /usr/bin/gs -q -sPAPERSIZE=a4 -dSAFER\
     -sDEVICE=epson -sOutputFile=- -
```

The above example assumes that the printer is an Epson-compatible device. Another type of printer could be substituted by adjusting the `sDevice` parameter.

Resourceful Linux users have contrived intelligent shell scripts that automatically recognize a file's format and then invoke the appropriate filter. Since such automatic recognition does not always work unambiguously, we refrain from giving further details.

6.4 Streamers and CD-ROMs

Many PC configurations today include a streamer and/or a CD-ROM drive. Linux supports such mass storage media, too. When purchasing a new streamer or CD-ROM drive, choosing a SCSI device simplifies Linux installation. Due to the standardization of

SCSI

the command set for SCSI devices, configuration proves quite easy. If the SCSI driver has already been compiled into the kernel, it suffices to make the appropriate entries in the `/dev` directory via the command `mknod`. Most Linux distributions create these device files automatically during installation.

For example, to configure CD-ROM drives (scd?) and SCSI streamers (rmt?), the following entries must exist:

```
linux1:/dev> ls scd* rmt*
crw-rw-rw-   1 root     root       9,   0 Jan 23  1993 rmt0
crw-rw-rw-   1 root     root       9,   1 Jan 23  1993 rmt1
brw-rw-rw-   1 root     root      11,   0 Jan 23  1993 scd0
brw-rw-rw-   1 root     root      11,   1 Jan 23  1993 scd1
linux1:/dev>
```

files in /dev

If these entries do not exist, the system administrator can generate them with the following commands:

```
linux1:/dev>mknod /dev/rmt0 c 9 0
linux1:/dev>mknod /dev/scd0 b 11 0
```

other drivers

Because of the lack of standards for other drivers, the configuration of a floppy streamer or a CD-ROM drive with its own IDE controller can prove significantly more problematic. However, corresponding drivers for the most common models have become available under Linux; to be used, these must be compiled into the kernel.

6.5 Network configuration

TCP/IP

Anyone with access to a TCP/IP network, whether through a university with Internet connection, at a company, or at home, can connect a Linux PC to this network. The following subsections describe the procedure in detail.

Addresses

IP address

In a network connection with TCP/IP, each network interface of a computer receives an IP address that is unique worldwide. The address consists of four numbers between 0 and 255, separated by

periods, for example, 141.7.1.40. On startup of the computer, this address must be set in one of the `rc` scripts in the `/etc/rc.d` directory.

network mask, router, nameserver

Other important data necessary for the configuration of the network are the network mask, broadcast address, the address of the router, and possibly the address of a name server. These terms are explained briefly in the following subsections. However, a detailed discussion of the administrative details would be beyond the scope of this book. Thus in the following we deal only with the practical effects of these parameters and refer the interested reader to the many books dedicated to TCP/IP and network administration.

RFCs

The exact definition of the protocols and concepts can be taken from the RFCs (Requests for Comments), which can be downloaded as text files from any larger ftp server.

Networks usually have a network administrator who manages the addresses and knows all important parameters. For small private networks that lack connections to other networks, the IP address is relatively insignificant. The user(s) can select them at will or assume the default values suggested by the installation program. Here it is only important to ensure unambiguity within the local area network.

network classes

When a direct connection to the Internet is being set up, the machine is assigned a network address in one of the classes A, B, or C. Depending on its class, this network address consists of one, two, or three bytes. The first bits of an address indicate the class to which the address belongs.

network address

The network address constitutes the start of the IP address. The local network administrator can assign the remaining bits. In a class C network the IP addresses consist of an officially determined network address with 3 bytes and a local component with 1 byte. Since addresses ending with 0 or 255 have a special meaning, a class C network can encompass up to 254 IP addresses.

subnetworks

The local component of the IP address can either be used directly for addressing the individual machine or divided into a subnetwork address and a host address by means of a network

mask. However, we do not further discuss the use of subnetworks here. Details can be found in rfc950.

In the following let us assume a class C network without subnetworks. The network mask for such a network is 255.255.255.0. This means that 3 bytes are used for network addressing and the remaining byte for host addressing.

The network address is entered with a trailing zero. In our example the network address might be 193.48.121.0 and the IP address of the Linux machine 193.48.121.37.

broadcast address

Another parameter needed for the configuration of the network is the broadcast address. TCP/IP uses a broadcast to handle certain tasks. In the simplest case and in our example, this is the network address with a final 255 instead of 0, i.e. 193.48.121.255.

For other computers on this network, only the last number of the IP address should be changed and only numbers from 2 to 253 should be used. Number 1 or 254 is normally used for a router and 0 and 255 for the network and broadcast addresses, respectively.

Loopback device

virtual interface

Even for computers that do not actually possess a network interface, there is a virtual interface named *loopback*. As its name suggests, everything that is output via this interface is input again directly. This allows TCP/IP connections for a machine to itself. Thus the TCP/IP programs can be used even without a network interface. The IP address of the loopback interface is usually 127.0.0.1.

Configuration of network interfaces

ifconfig

The `ifconfig` command provides the means to configure a computer's network interface and thus to determine its IP address and the other parameters mentioned above. The command takes as its parameters the interface to be configured, the IP address, and other optional parameters. Details about these parameters can

be found on the on-line reference manual page for `ifconfig`. The names of the interfaces are displayed by the drivers during booting. Examples of usual names include `eth0` for Ethernet, `lo` for loopback, and `sl0` for the first serial interface with SLIP.

Routing

route

Once the interfaces have been configured, entries must be made in the internal routing table of the kernel. The routing table stores which addresses are accessible via which interfaces. The command `route` supports making these entries.

The following network interface configuration example shows the script `/etc/rc.d/rc.inet1`, which carries out the necessary settings for a computer with IP address 193.48.121.37 in network 193.48.121.0.

```
HOSTNAME=stef1

# Activate loopback
/sbin/ifconfig lo 127.0.0.1
/sbin/route add  net 127.0.0.0

# Parameters for this computer:
IPADDR="193.48.121.37"
NETMASK="255.255.255.0"
NETWORK="193.48.121.0"
BROADCAST="193.48.121.255"
GATEWAY="193.48.121.1"

# Activate Ethernet interface
/sbin/ifconfig eth0 ${IPADDR} broadcast ${BROADCAST} netmask
${NETMASK}

# Routing for the local network
/sbin/route add  net ${NETWORK} netmask ${NETMASK}

# Routing for other addresses via the router
/sbin/route add default gw ${GATEWAY} metric 1
```

SLIP

modem

Normally, SLIP connections are not configured at system startup, but only when they are actually needed. The SLIP protocol is used to make TCP/IP connection via a modem or any serial line. The configuration takes place only after a telephone connection has been established with the SLIP server.

The program `dip` is for dialing the telephone number and making the connection to a SLIP router or server. It can be used interactively or controlled from a script. Once the connection has been established with `dip`, the SLIP interface (sl0) is activated and configured as the default gateway in the routing table of the kernel.

The following example is a simple script for dialing a Linux SLIP server. It is invoked with `dip <scriptname>`.

```
# DIP - Login Script  (for dip 3.3.7)

main:
  # determine IP-addresses
  get $local opcon.franken.de
  get $remote wuff.mayn.sub.de

  # configure port
  port cua1
  speed 38400
  reset

  # initialize modem
  send ATQ0V1E1X4\r

  # wait for OK
  wait OK 2
  if $errlvl != 0 goto modem_trouble

  # dial telephone number
  dial 993322
  if $errlvl != 0 goto modem_trouble

  # wait for CONNECT
  wait CONNECT 60
  if $errlvl != 0 goto modem_trouble

login:
  sleep 2

  # wait for login message
  wait ogin: 20
  if $errlvl != 0 goto login_error

  # send login name (linux)
  send linux\n

  # wait for password prompt
  wait ord: 20
  if $errlvl != 0 goto password_error

  # send password (linus)
  send linus\n

loggedin:

  get $mtu 296
  default

done:
  print CONNECTED $locip ---> $rmtip
  mode CSLIP
  goto exit
```

```
prompt_error:
  print TIME-OUT waiting for SLIPlogin to fire up...
  goto error

login_trouble:
  print Trouble waiting for the Login: prompt...
  goto error

password:error:
  print Trouble waiting for the Password: prompt...
  goto error

modem_trouble:
  print Trouble ocurred with the modem...
error:
  print CONNECT FAILED to $remote

exit:
```

Ping

testing connections

One of the most important tools for troubleshooting a UNIX network is the command `ping`. After the network interface has been configured with `ifconfig` and the necessary routing information has been set, the connection can be tested. `Ping` sends small data packets at regular intervals to the target machine, which then responds.

```
linux0:/home/tul> ping linux1
PING linux1.openconcepts.com (192.0.2.130): 56 data bytes
64 bytes from 192.0.2.130: icmp_seq=0 ttl=119 time=2 ms
64 bytes from 192.0.2.130: icmp_seq=1 ttl=120 time=1 ms
64 bytes from 192.0.2.130: icmp_seq=2 ttl=121 time=1 ms
64 bytes from 192.0.2.130: icmp_seq=3 ttl=122 time=1 ms
64 bytes from 192.0.2.130: icmp_seq=4 ttl=123 time=1 ms

--- linux1.openconcepts.com ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 1/1/2 ms
linux0:/home/tul>
```

This also allows the assessment of the transmission speed. Network configuration should proceed only after the connection has been tested successfully with `ping`.

Host names and name server

Most users do not consider the direct input of an IP address to be a very comfortable solution. This motivated the introduction of

symbolic addresses. Such a symbolic address consists of the host name and the domain name.

The host name is a locally unique designation for a machine. The domain name, on the other hand, must be internationally unambiguous. Instead of the IP address 152.2.22.81, for example, a user could enter the symbolic address `sunsite.unc.edu`. In this case `sunsite` is the host name and `unc.edu` the name of the domain.

host name, domain name

The file `/etc/hosts` establishes the correspondence locally of IP address and symbolic address. Here additional names, or aliases, can be defined for a host. Aliases serve to define shorter names and thus to make the entry of addresses more comfortable. The following is an excerpt from the file `/etc/hosts`:

alias

```
#IP address    symbolic address              alias

141.7.21.40     linux1.test.fh-heilbronn.de     linux1

127.0.0.1      localhost

141.7.21.20     sun1.test.fh-heilbronn.de       sun1
141.7.21.25     risc1.test.fh-heilbronn.de     risc1      news
```

Since maintaining hundreds or thousands of entries in this file proved impossible, a hierarchical system of name servers was set up to manage these symbolic names and automatically exchange addresses among themselves. For each domain, i.e. all machines with a certain domain name, there is a responsible name server. A detailed description of the concepts domain name, subdomain, and name server can be found in rfc1034 and rfc1035 (also see page 26).

name server

The UNIX TCP/IP programs convert the host name to the corresponding IP address by invoking a C library routine. This routine, called *resolver*, reads the file `/etc/hosts` and makes a connection to the next name server if necessary. The order in which this is to occur can be specified in the file `/etc/host.conf`.

resolver

```
order hosts, bind
multi on
```

The entry `order hosts bind` indicates that the file `/etc/hosts` is to be read first. If it contains no entry for the host name in question, then the name server is contacted. The address of the next name server is stored in the file `/etc/resolv.conf`. The line `multi on` means that the resolver should return all valid addresses to a host when multiple addresses in are entered in the file `/etc/hosts`. The on-line reference manual page for `resolv+` describes these and other options in detail.

Many Linux installation packages include name server software. The configuration of this name server is no simple matter, however, and for smaller networks a name server usually proves superfluous. For detailed descriptions of how a name server functions and of TCP/IP configuration files, refer to the *Linux Network Administration Guide* (NAG), the RFCs mentioned above, or specialized books about TCP/IP.

E-Mail configuration

smail

The files in the directory `/usr/lib/smail` contain the configuration of the smail daemon, which is responsible for sending and receiving mail. In the simplest case, the following entries in the file `config` suffice.

```
#
#  Configuration of smail in a LAN
#

# domain name
visible_domain=rz.fh-heilbronn.de

# mail address of host (can differ from the host name)
visible_name=test.rz.fh-heilbronn.de
```

UUCP

If mails are to be sent not just within a LAN (Local Area Network) but via a UUCP connection to the outside as well, then the computer (linux2) with the UUCP connection is defined as `smart_path`.

```
#
#  Configuration of smail in a LAN with UUCP connection
#

# Domain name
visible_domain=rz.fh-heilbronn.de

# Mail name of host (can differ from host name)
visible_name=test.rz.fh-heilbronn.de

# Routing via UUCP computer (linux2)
smart_path=smartie.rz.fh-heilbronn.de
```

To allow it to forward the mails of the other computers, the corresponding UNIX UUCP computer is then configured as follows:

```
#
#  Configuration of smail in a LAN with UUCP connection
#

# Domain name
visible_domain=rz.fh-heilbronn.de

# Mail name of host (can differ from host name)
uucp_name=linux2.rz.fh-heilbronn.de
visible_name=linux2.rz.fh-heilbronn.de

# Routing via UUCP computer (linux2)
smart_path=janus
smat_transport=uux
```

More detailed information on configuration can be found in the on-line reference manual pages corresponding to `smail` and in the *Linux Network Administration Guide* (NAG).

Inetd

The `inetd` daemon manages Internet services. It waits for requests for connections to port numbers that have been specified for Internet services and starts the corresponding daemon only upon an actual request for a connection. The file `/etc/inetd.conf` contains the table that specifies which daemon is responsible for which services.

Internet services

```
telnet  stream  tcp  nowait  root  /etc/telnetd    telnetd
ntalk   dgram   udp  wait    root  /etc/ntalkd     ntalkd
ftp     stream  tcp  nowait  root  /etc/ftpd       ftpd -l
finger  stream  tcp  nowait  root  /etc/fingerd    finger
shell   stream  tcp  nowait  root  /etc/rshd       rshd
login   stream  tcp  nowait  root  /etc/rlogind    rlogind
tftp    dgram   udp  wait    root  /etc/tftpd    tftpd echo
echo    dgram   udp  wait    root  internal
discard stream  tcp  nowait  root  internal
discard dgram   udp  wait    root  internal
daytime stream  tcp  nowait  root  internal
daytime dgram   udp  wait    root  internal
chargen stream  tcp  nowait  root  internal
chargen dgram   udp  wait    root  internal
```

port numbers

The file `/etc/services` establishes the correspondence of services to port numbers.

```
tcpmux          1/tcp       # TCP Port Service Multiplexer
rje             5/tcp       # remote job entry
echo            7/tcp
echo            7/udp
discard         9/tcp       sink null
discard         9/udp       sink null
systat          11/udp      users
systat          11/tcp      users
daytime         13/udp
daytime         13/tcp
daytime         13/udp
netstat         15/udp
netstat         15/tcp
qotd            17/udp      quote
quote           17/tcp      # quote of the day
chargen         19/tcp      ttytst source
chargen         19/udp      ttytst source
ftp data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp      mail #Simple Mail Transfer
nsw-fe          27/tcp      # NSW User System FE [24, RHT]
```

These files normally remain unchanged. Their modification becomes necessary only when new services are added or when the update of a daemon involves new options.

Berkeley r-Utilities

The Berkeley r-Utilities `rlogin`, `rsh` and `rcp` also have their own configuration files. In order to be granted remote access to another computer with these programs, the client machine must be entered in the file `/etc/hosts.equiv` on the server. A user name can be entered along with the host name in this file. This restricts access to this specific user on the specified host.

remote access

The daemon `lpd` also uses the file `hosts.equiv`, but also searches for permissions in the file `/etc/hosts.lpd`, which has

the same format as the file `hosts.equiv`. If a client only needs access to a printer queue, then for security reasons the name of the client should be entered only in the file `/etc/hosts.lpd`.

The following demonstrates the format of the files `hosts.equiv`, `hosts.lpd` and `.rhosts`.

```
#
# Valid hosts and users
#

linux1.rz.fh-heilbronn.de
linux2.rz.fh-heilbronn.de          strobel
sun1.rz.fh-heilbronn.de            arnold
```

Only the system administrator can modify the files `/etc/hosts.equiv` and `/etc/hosts.lpd`. If an individual user wants to grant permission to another user, an entry in the file `.rhosts` in the granting user's home directory suffices. This approach proves especially practical when a user has different user IDs on multiple machines on the network. Corresponding modification of the `.rhosts` file significantly eases access to the user's own directories throughout the network.

.rhosts

NFS and mount daemons

The Network File System (NFS) requires several daemons. Since NFS is based on Remote Procedure Calls (RPC), the portmap daemon is needed first. It registers the RPC services of a server and returns the TCP/IP port number to the client submitting requests.

portmap daemon

The NFS daemon `rpc.nfsd` replies to the read and write requests from NFS clients. The daemon `rpc.mountd` is responsible for the mounting itself; it manages directories and checks the permissions of a client submitting a mount request.

This daemon's most important configuration file is `/etc/exports`, which lists all directories that can be mounted by other machines via NFS along with their permissions. Modifications in this file become effective only after both `rpc.nfsd` and `rpc.mountd` are restarted.

The following is an example of this file:

```
#
# Exported directories
#
/               linux1(rw)
/home           141.7.1.49(rw)
/home/prog      risc1(rw)
```

6.6 X Window System configuration

The installation of the X Window System Version 11 (XFree86 2.1.1) under Linux normally consists of simply unpacking the programs and files from the various tar archives. With most Linux packages this occurs during the installation of the operating system, and so it poses no problem.

The configuration task is complicated when the X server must be adapted to the available video adapter and monitor. Then configuration requires modifying the central configuration file, which is usually in the directory `/usr/lib/X11/Xconfig`. According to the Linux File System Standard, however, this file should reside in `/etc/X11`.

Xconfig

The Xconfig file

This file starts with the definitions of the paths that the X server needs. These settings have usually been undertaken correctly in the various installation packages. Here is an excerpt showing the format of the `Xconfig` file:

```
# File with color definitions
RGBPath         "/usr/X386/lib/X11/rgb"

# Paths for fonts
FontPath    "/usr/lib/X11/fonts/Type1/"
FontPath    "/usr/lib/X11/fonts/75dpi/"
FontPath    "/usr/lib/X11/fonts/Speedo/"
FontPath    "/usr/lib/X11/fonts/BitstreamType1/"
FontPath    "/usr/lib/X11/fonts/misc/"

# Keyboard
Keyboard
  AutoRepeat 500 5
#  Xleds      1 2 3
  ServerNumLock
#  DontZap

LeftAlt         Meta
RightAlt        ModeShift
ScrollLock      ModeLock

# Maus
Microsoft       "/dev/mouse"
#MouseSystems   "/dev/mouse"
#MMSeries       "/dev/mouse"
#Logitech       "/dev/mouse"
#MouseMan       "/dev/mouse"
#Busmouse       "/dev/mouse"
```

Then the functions of special modifier keys need to be defined. Next comes the configuration of the system for the mouse by entering its type and the serial port it uses (see terminal capture above).

Then configuration information is entered for the VGA board, such as its chipset and the available driving clock frequencies (dot clocks) of the video adapter. This specification can be omitted for simple VGA boards since the X server detects the installed chipset on startup and the driving clock frequencies can also be determined automatically. However, it makes sense to specify these anyway, because the detected driving clock frequency is used as an identifier to which the definitions of the video modes later refer. If there were fluctuations during the detection of the driving clock frequency and a clock were identified at 49.5 Hz instead of 50, then the X server might not identify the frequency used by a video mode and thus terminate with an error message.

detecting dot clocks

To establish the available driving clock frequencies, the `clocks` line can be removed from the `Xconfig` file and the X server started with the option `-probeonly`.

```
# X -probeonly
```

The X server then displays the detected driving clock frequencies and other driver information in text mode and then quits.

The names of the video modes that are to be available at run time are listed in the line beginning with `Modes`. With the X server running, keys `<Ctrl-Alt +>` and `<Ctrl-Alt ->` of the numerical keypad allow switching the resolution among the available modes, which still need to be defined more exactly, however.

Setting the video modes

danger!

The most difficult and dangerous part of the configuration is the setting of the video modes, since this directly defines the synchronization frequencies (timing) of the monitor, and a fixed-frequency monitor or other monitor without protective circuitry against exceeding its bandwidth can be damaged by incorrect values. The `Xconfig` file included in the XFree86 distribution contains the most important video modes that should work on modern monitors without causing damage.

The advantage of this kind of configuration of the video mode is that the available monitor can be used to full advantage. For example, a 14" monitor whose maximum horizontal synchronization frequency (HSF) is too low to display 800 x 600 pixels flicker-free could be operated at a resolution of 800 x 550 with 72 Hz screen refresh rate (RR).

For each mode, the file `/usr/lib/X11/Xconfig` specifies the clock to be used as well as four values each for horizontal and vertical synchronization. This is a sample definition of a video mode:

```
# Mode      Clock     horizontal                vertical
  "800x600"  45       800  840 1030 1184       600  600  606 624
```

video mode values

The respective meanings of the values in each group of four numbers are:

- the maximum number of pixels after which the picture ceases to be displayed
- the number of dot-clock ticks to the start of the horizontal synchronization pulse (sync), whereby the values are counted ongoing
- the number of pixel steps until sync is ended and the second guard time of the electron beam begins
- the total number of pixel steps to the end of a cycle (frame)

```
Modes         "800x5" "800x600"
Clocks      25 28 45 40 33 29
Chipset      "et3000"
Videoram    512

ModeDB

#              clock  horizontal timing      vertical timing
 "800x600"     40     800  800  860 1030     600  604  610  624
 "800x5"       45     800  840 1030 1120     540  540  546  558
```

The above example defines three video modes. The video adapter has an ET3000 chip and 512 KB of RAM. The clocks are defined as fixed and are assigned names, so that the quartz with name 45 activates the third available driving clock frequency. The line with "800x5" means that a video mode with the name "800x5" is defined for which a clock with the name 45 is to be used.

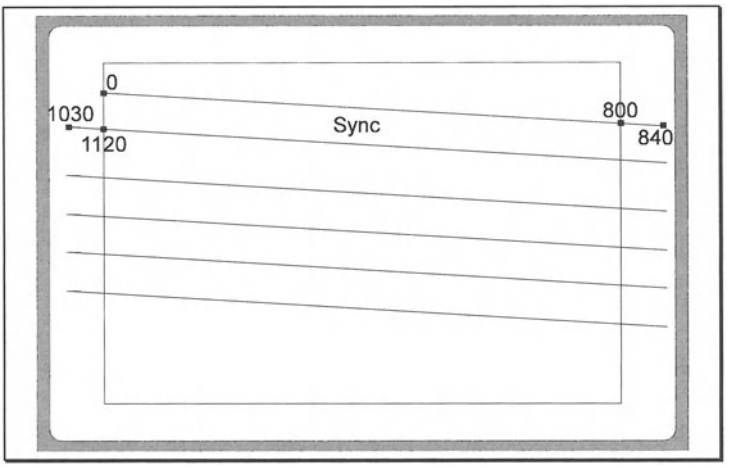


Fig. 6.1 Schematic representation ot the composition of a screen

The horizontal resolution is 800 pixels, and after the end of a visible line a guard time begins for the electron beam to rest. The guard time lasts until 840 pixels; then the synchronization pulse begins. It lasts 190 dot-clock ticks until 1030. Then there is a guard time until 1120. Thereupon the next horizontal cycle begins.

After 540 horizontal cycles (that is, scan lines) the vertical synchronization intervenes, lasting 6 horizontal cycles. Then there is another guard time until the 558th cycle. After the guard time, the next screen is ready to start.

determination of values

The file `VideoModes.doc`, which is included in the file `XF86VidDoc.tar.gz` of the XFree86 distribution, describes in depth the rules for determining the exact values for such a video mode. One source for this file is the ftp server `sunsite.unc.edu` of the University of North Carolina in the directory `/pub/Linux/X11/XFree86-2.1.1/`. However, it is often simpler to find a corresponding entry in one of the many example files and to modify it.

There is also a table for the simple spreadsheet program `sc`. This can be found on the ftp server `sunsite.unc.edu` and its mirror servers in the directory `/pub/Linux/X11/install` with the file name `modegen.taz`.

horizontal sync frequency

The limiting factor for simple monitors is usually the maximum horizontal synchronization frequency. This is the frequency with which the electron beam moves from left to right and from scan line to scan line. This frequency is computed by dividing the driving clock rate, specified in MHz, by the largest (right) number of the block for horizontal timing. The following formula computes the horizontal synchronization frequency:

$$f_{horizontal} = \frac{f_{pixel}}{N_{pixel}}$$

In the above example the required horizontal sweep frequency would be 45 MHz / 1120, or approximately 40 kHz. This also happens to be the upper limit for the monitor in our example.

Vertical synchronization frequency

To compute the vertical synchronization frequency (vertical timing), divide the horizontal synchronization frequency by the number of scan lines (i.e., horizontal cycles) necessary for a complete screen. This is the rightmost number in the block for vertical timing. The following shows the formula for computing the vertical sweep frequency.

$$f_{vertical} = \frac{f_{horizontal}}{N_{lines}}$$

In our example we have 40 kHz / 558, or 72 Hz. If 540 lines rather than 600 are to be displayed, then the vertical synchronization frequency falls significantly below 72 Hz, which the user notices as light flickering of the monitor.

flickering

With a better monitor whose maximum horizontal synchronization frequency is, for example, 60 kHz, and a newer video adapter that offers a faster clock, the driving clock frequency could be raised to achieve a higher vertical synchronization frequency.

To modify an existing video mode, we recommend copying and modifying the mode repeatedly and entering the modified modes in the mode lines with different names. Then start the X server and compare the effects of the modifications by switching modes with `<Ctrl-Alt>` and the `+` or `-` key on the numerical keypad. If the monitor no longer synchronizes with a new mode, i.e., it fails to show a stable picture, quickly change modes or end the X server with `<Ctrl-Alt-Backspace>` to avoid damage to the monitor.

The program `vgaset` provides a valuable aid in adjusting the picture. Started in an `xterm`, it permits interactive manipulation of the picture position. At the touch of a key the borders can be increased or decreased, and the duration of the synchronization signal can be changed. The eight values to be entered for the current settings in the file `Xconfig` are constantly displayed.

vgaset

Keyboard layout configuration

The X Window System manages the keyboard independently of the kernel. An American keyboard is initialized as the default. Country-specific keyboard tables can be loaded with the utility

xmodmap

`xmodmap`. When the X server is launched in a normal configuration, `xmodmap` is invoked with the file `.Xmodmap`. The utility seeks this file first in the user's home directory and then in the directory `/usr/lib/X11/xinit`.

If a given keyboard layout is to be defined system-wide, it must be written in the directory `/usr/lib/X11/xinit`. Ready `.Xmodmap` files are included in some distributions, or they can be drawn from ftp servers such as `sunsite.unc.edu` in the directory `/pub/Linux/X11/misc`.

Since `xmodmap` is normally invoked in an `xinitrc` script, the invocation might need to be modified to seek the file `.Xmodmap` in another directory or under another name. In case of doubt, the script that starts the X server, usually `startx`, should be examined. In this script `xinit` is invoked with the names of additional scripts, normally including `.xinitrc` in the user's home directory or in the directory `/usr/lib/X11/xinit`.

xkeycaps

Using the `xkeycaps` utility certainly proves to be a more comfortable alternative. It is included in many Linux distributions and can be found on the usual ftp servers. This program affords an X Window System user interface. The user can display the current keyboard layout and interactively change it with the mouse.

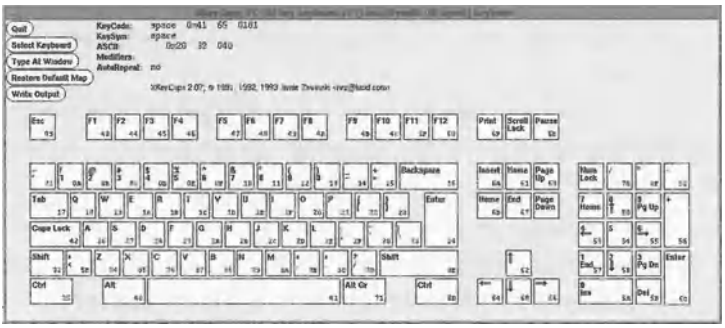


Fig. 6.2 Interactive reconfiguration of the keyboard layout

Administration

Upon completion of the installation of the Linux system and the most important aspects of configuration, it is time to enable other users to access the system. In addition, there will soon be requests for applications that are not contained in the installation packages. Over time new versions of some system components will appear and need to be installed to keep pace with continuing development. These kinds of jobs are collectively termed *system administration*. Since system administration tasks are very similar on all UNIX systems, we refer the reader to the standard literature and only share some tips and notes on Linux-specific details.

system administration

7.1 The Administrator

Only the user *root*, that is, the system administrator, can modify the configuration files of the system. The corresponding permissions of these files guard them against unauthorized use. The system administrator generally has access to all files and can modify these at will.

root

This also means that the administrator could crash or erase the entire system. For example, imagine that an administrator were to carelessly enter the following command to delete all files in a directory (with options set to recursively remove all subdirectories, and without confirmation) and do to so from the root directory (/):

```
linux1:/> rm -rf *
```

This would immediately delete the entire system. If this command were to be executed without root permissions, then the access privileges of the system files and directories would assure that at worst all the user's own files would be deleted, and this would have no influence on other users or the overall system.

Thus a system administrator must proceed with extreme caution and only log in to the *root* if a system file needs to be modified or a new program is to be installed.

7.2 Booting

To facilitate understanding of the system, we next describe how a Linux system boots and which programs and scripts are processed during booting.

master boot record

Independent of the operating system the *master boot record* (MBR) is loaded first during booting. The partition tables and the loader program that loads the boot sector of the active partition are located here.

LILO

If the Linux Loader (LILO) is installed in the MBR as described in Chapter 5, then it is started first and presents a choice of two Linux kernels and DOS. If the user selects a Linux kernel, the respective kernel image is loaded and started. This represents the actual start of the Linux system.

kernel

First the kernel initializes the graphic board and possibly prompts for the desired screen resolution. Then it installs the various device drivers, which usually output a comment on the

init

console. Then the kernel mounts the root file system and starts the process `init`.

Like UNIX System V, Linux recognizes various run levels, which are specified in the file `/etc/inittab`. These are various configurations in which only certain system components are activated. Normally the system starts in multi-user operation.

multi-user

This means that multiple `getty` processes are started for the console and optionally for the serial ports. In addition, in this mode all network daemons are activated. Single-user mode provides an alternative that is intended primarily for system

administration. Another run level might start a graphic login prompt (xdm) instead of the usual terminal login, for example.

Because init is the first process that the kernel starts, it is always assigned process number 1 and is the parent of all further processes. The init process also executes various scripts in the directory /etc/rc.d or /etc, which usually begin with rc. These scripts reinitialize system files and mount the local file systems. NFS file systems are mounted later because the network and the respective daemons have not been started. The exact sequence and ordering of the various scripts can vary from system to system. The following description is intended as an example to explain the principles.

script

The mount command, which grafts additional file systems into the Linux directory tree, reads the file /etc/fstab, in which all file systems are listed along with their device, type, and mount point (the directory to which the file system is to be attached). If a new hard disk is added or other new file systems become available, these need to be added to this file. This is the script /etc/rc.d/rc.M

mounting file systems

```
#! /bin/sh
#
# rc.M              This file is executed by init(8) when the system is being
#                   initialized for one of the "multi user" run levels (i.e.
#                   levels 1 through 6). It usually does mounting of file
#                   systems et al.
#
# Version:          @(#)/etc/rc.d/rc.M     2.02       02/26/93
#
# Author: Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#

  # Tell the viewers what's going to happen...
  echo "Going multiuser..."

  # Start update.
  /sbin/update &

  # Screen blanks after 15 minutes idle time.
  /bin/setterm -blank 15

  # Initialize the NET subsystem.
  if [ -x /etc/rc.d/rc.inet1 ];
  then
        /bin/hostname linux2
        /bin/domainname rz.fh-heilbronn.de
        /bin/sh /etc/rc.d/rc.inet1
        /bin/sh /etc/rc.d/rc.inet2
  else
        /sbin/hostname_notcp linux2
        /bin/domainname rz.fh-heilbronn.de
```

```
        echo
        echo "Since you don't have TCP/IP installed, syslogd will complain"
        echo "when it first starts. The warning can be ignored."
        echo
          /usr/sbin/syslogd
          /usr/sbin/klogd
          /usr/sbin/lpd
          /usr/sbin/crond
  fi

  # Remove stale locks (must be done after mount -a!)
  /bin/rm -f /usr/spool/locks/* /usr/spool/uucp/LCK..* /tmp/.X*lock 1> /dev/null
2> /dev/null

  # Remove stale hunt sockets so the game can start.
  if [ -r /tmp/hunt -o -r /tmp/hunt.stats ]; then
    echo "Removing your stale hunt sockets from /tmp..."
    /bin/rm -f /tmp/hunt*
  fi

  # Update all the shared library links automatically
  /sbin/ldconfig

  # Start the local setup procedure.
  /etc/rc.d/rc.local

  # All done.
```

If the configuration includes a network connection, this file starts the shell scripts `rc.inet1` and `rc.inet2` to initialize the network environment. Any NFS mounts that have been defined are executed here as well.

daemons

7.3 Shutdown

As with all UNIX systems, a Linux computer must not simply be turned off. Instead the system must be shut down with the command `shutdown`. The reason for the necessity of the shutdown procedure is that, due to the internal cache of the kernel, usually all data that were written by programs to the hard disk interface have not yet physically been stored on the hard disk. Furthermore, frequently needed information, such as the i-node table and the superblock of the file system, are likewise held in RAM. Turning off the computer without using the command `shutdown` can lead to inconsistencies on the hard disk and resulting data losses. The command `shutdown` assures that all buffers have been transferred to the st
orage media and that all processes are terminated properly. (The command `sync` writes buffers to the hard disk without shutting down the system.)

disk cache

7.4 The Linux directory tree

To assist new Linux users and inexperienced system administrators in gaining an orientation in the system, this section describes the most important directories of a typical Linux system. The organization of the Linux file system has been specified in the Linux File System Standard (FSSTND), which resides as a PostScript file on the usual ftp servers along with other documents. This standard has been recognized by most of the producers of distributions and packages, and the following description assumes a typical distribution.

Linux File System Standard

The directory / is the root of a Linux directory tree. It is thus called the root directory. Beyond the Linux kernel image files that are needed for booting and the most important subdirectories, it should contain no other files.

root directory

System administrators frequently use the root directory as their home directory. However, we recommend creating a separate directory for this purpose, e.g., `/root`. This makes it easier to distinguish the administrator's configuration files from system files.

/root

Directories in the root directory

- **`/etc`** - The `/etc` directory contains local configuration files, including the files `passwd` and `group` with the user and group information, respectively, and the configuration files for the TCP/IP daemon, such as `services`, `inetd.conf`, and `exports`. Before the file system standard, daemons and system programs such as `init` and `update` were stored here. These now reside in `/sbin`.

configuration files

- **`/etc/skel`** - On creation of a new user with the command `useradd -m`, the files in this directory are automatically copied to the user's home directory. Examples of user-specific configuration files are normally stored here, including files like `.cshrc`, `.bashrc`, `.Xdefaults`, and `.emacs`.

new user

- **`/etc/keytables`** - The file system standard assigns this directory for keyboard layout tables that can be loaded upon booting. American distributions sometimes use the directory

keyboard

/usr/lib/keytables or /usr/lib/kbd/keytables for this purpose.

scripts

- **/etc/rc.d** - The scripts invoked by init on booting the system usually reside in this directory. Alternatively, they can reside in the /etc directory.

X11 configuration files

- **/etc/X11** - The file system standard uses this directory for X11 configuration files, including Xconfig with its general settings for the server and the monitor, Xmodmap with the keyboard layout under X11, and xinitrc.

configuration

- **/conf** - If this directory exists at all, it contains exclusively configuration files which would otherwise be found in /etc or other directories. In this case /etc contains not the actual files themselves but only symbolic links to the files in /conf or a subdirectory thereof, such as /conf/net. In simple installations, however, this directory is usually omitted.

devices

- **/dev** - This directory encompasses device drivers, which are special files that correspond to an input/output driver (See also Section 2.6).

temporary files

- **/tmp** - Many programs use this directory for temporary files. All users can read from and write to /tmp. Files in this directory can normally be deleted when no application processes are running, but no user other than the administrator should be logged in during the deletion procedure. Many system administrators delete the files in the /tmp directory on system startup by means of an entry in the file /etc/rc.d/rc.local.

mounting

- **/mnt** - This directory should be empty. It is often used to temporarily mount diskettes or remote file systems per NFS.
- **/user** - Likewise this directory is normally empty, if it exists at all, and is used for mounting.

kernel information

- **/proc** - Normally the proc file system is mounted here, a special file system in which information about the kernel and running processes is represented as subdirectories and files. These files can usually be read as text and thus permit easy access to this information.

system administration

- **/sbin** - This directory contains only the most important programs and commands needed for booting the system and for basic system administration. These include getty, init,

update, fdisk, fsck, ifconfig, ping, and lilo. Programs used by users other than root reside in /bin or in /usr/bin if they are not absolutely necessary.

- **/bin** - The most important system programs, specifically those that have to be present if the directory /usr proves inaccessible, reside in this directory. This includes the commands mv, cp, cat, and rm. Unlike /sbin, which contains only vital programs for system administration and for booting, /bin contains programs intended for all users. All other commands needed in case of an emergency reside in /usr/bin (see also /usr). — administration
- **/lib** - The images of the shared libraries of the system are located in the directory /lib. This is the part of a shared library that contains the actual routines and that is loaded with the starting of a program that uses the library. The other part, called the *stubs*, is stored in the directory /usr/lib; they are linked to the programs and contain only references to the actual routines. Shared libraries that are not absolutely needed for booting and for administration, such as the libraries of the X Window System, should be placed in another subdirectory under /usr, this being /usr/X386/lib in the case of X11. /lib only contains symbolic links for these libraries. — shared libraries; stubs
- **/home** - In this directory a home directory is created for every user except root. Each user's respective directory then contains user-specific configuration files. Apart from the user's personal files, no other programs should be installed here. — home directories

  Since this directory is usually located on a separate partition, we do not recommend creating the home directory of root in this directory as well. If this file system should become unmountable due to an error, then even the administrator might not be able to log in to the system to correct the error. — warning
- **/install** - The installation programs of some distributions use this directory to store information about installed packages. Other distributions use special directories under /var or /usr.
- **/boot** - Map files of the Linux Loader and backup copies of all boot sectors and of the partition table reside here. These

files are normally used only by LILO or automatically created by LILO.

- **/var** - This directory contains all files that are written to often and whose size changes frequently. This includes primarily log and spool files. Many subdirectories of /var previously resided under /usr. To be able to mount /usr as read-only per NFS from multiple computers simultaneously, these dynamic subdirectories were relocated in the /var directory. Such subdirectories include /var/spool with the subdirectories for mail and the printer queue, /var/adm with the system log files, and /var/lock with the lock files.

log and spool files

- **/usr** - This directory contains almost all other important directories that are not necessary for the systems startup. The separation of machine-independent configuration, essential programs for system administration, and log and spool files from the programs in /usr is intended to enable using the directory /usr for multiple machines from a single NFS server. To avoid conflicts, this requires mounting the /usr directory write-protected.

read only

The most important programs for system administration and the necessary libraries must reside in the root file system, however, so that if a system error makes it impossible to mount the NFS server, the error can be corrected. The root file system should be as small as possible to allow shared use of as many programs and as much disk space as possible.

Subdirectories under /usr

shared configuration

- **/usr/etc** - This directory should contain the configuration files that can be used jointly by multiple machines. Often these amount to symbolic links to the directory /etc.
- **/usr/bin** - Most of the system programs and UNIX commands for users and for the system administrator that are not absolutely essential in case /usr cannot be mounted reside here. The separation of UNIX commands into those that should be in /bin or /sbin and those that should be in /usr/bin is not always carried out consistently. When

looking for a file, check both directories. Both `/bin` and `/usr/bin` should be part of the path (PATH).

- **`/usr/bin/X11`** - X Window System programs are normally installed in this directory. However, This is usually only a link to `/usr/X386/bin` — X11 programs
- **`/usr/lib`** - The static libraries for various programming languages and the stubs for the shared libraries reside here. In addition, this directory contains several subdirectories that usually contain auxiliary and configuration files of other programs. — libraries
- **`/usr/lib/X11`** - Here we find the configuration data, fonts, color tables, and other files of the X Window System. This directory is usually a link to `/usr/X386/lib/X11`. According to the file system standard, files that involve the local configuration of the X server, such as `Xconfig`, should reside under `/etc/X11`. All distributions do not adhere to this aspect of the standard. — X configuration
- **`/usr/include`** - The include files of the C library are stored here. This directory includes the subdirectories `sys` and `linux`, whereby `linux` is only a reference to a subdirectory of `/usr/src/linux`. — C includes
- **`/usr/g++-include`** - This directory encompasses the C++ include files. — C++ includes
- **`/usr/man`** - Manual pages are stored in subdirectories of `/usr/man`.
- **`/usr/info`** - The GNU Information System uses this directory. The files in this directory can be viewed in info mode in the `emacs` editor or with programs such as `tkinfo`. They represent the primary documentation of GNU programs. — info mode
- **`/usr/doc`** - Other documentation that is not available as a manual page or in info format resides in this directory.
- **`/usr/src`** - The subdirectories of `/usr/src` contain the source codes of system programs. The most important of these subdirectories is `/usr/src/linux`, which contains the source code of the Linux kernel. — source code
- **`/usr/local`** - This directory should serve as the target for the installation of programs that were not part of the installation package. As such, it usually contains a complete

subdirectory tree consisting of `bin`, `lib`, `etc`, `include`, and `man` directories. As a rule `/usr/local/bin` is part of the path for programs (PATH) and `/usr/local/man` for on-line reference manual pages (MANPATH).

- **`/usr/X386`** - This subdirectory begins the actual directory tree of the X11 package. The directories `/usr/lib/X11` and `/usr/bin/X11` are links to this directory tree.
- **`/usr/openwin`** - The subdirectories of `/usr/openwin` contain the programs and data of the Sun XView package. The libraries and configuration files are usually in the subdirectory `/usr/openwin/lib`. The most interesting of these files are the definition files of the menus for the OpenLook Window Manager (`olwm` and `olvwm`). Their file names all begin with `openwin-menu`. The window manager and the other programs are located in `/usr/openwin/bin`.

OpenLook

- **`/usr/TeX`** - The TeX package is installed in this directory. However, according to the file system standard the TeX data files should reside under `/usr/lib/TeX`.

TeX

7.5 Users and groups

Every user has a unique user ID and belongs to one or more groups. This information is stored in the files `/etc/passwd` and `/etc/group`.

To create a new user, these files are not usually modified with a text editor. Instead, the program `useradd` is used. This program collects all the important information via the command line. Then it modifies the files `/etc/passwd` and `/etc/group` as well as the respective *shadow files*, which contain the encrypted user and group passwords and, for security reasons, can only be read by the superuser.

useradd

shadow file

The administrator can simplify the task of managing users and groups by entering the `-D` option with `useradd` once to set default values for the group, longevity of the password, and the directory for the home directories of the users.

Furthermore, user-specific configuration files, such as `.profile`, `.bashrc` and `.openwin-menu` can be stored in

directory `/etc/skel`. When a new user is created, these files are automatically copied into the user's home directory.

If default values have been defined and the correct files have been stored in `/etc/skel`, then a new user can be created by invoking:

```
linux1:/> useradd -m <user name>
```

The user ID will automatically be the next free number.

password

Next, a password has to be assigned with `passwd <user name>`, since the new account would otherwise remain disabled.

Beyond `useradd` there are commands to modify the settings of a user, such as `usermod`, `userdel`, `chsh` to change the login shell, and `chfn` to change the full name. The on-line reference manual page for `useradd` gives references to these and other commands. The following example shows the specification of default values and the adding of a new user:

```
dirk1:/etc# useradd -D -g 6 -b /home -f 3 -e 999
dirk1:/etc# useradd -m peter
dirk1:/etc# passwd peter
Changing password for peter
Enter the new password (minimum of 5 characters)
Please use a combination of upper and lower case letters and
numbers.
New Password:
Re-enter new password:
dirk1:/etc#
```

Groups are managed with the commands `groupadd`, `groupmod` and `groupdel`. To assign a user to another group, the command `usermod` with the option `-G` is used.

Anonymous ftp

user ftp

Besides `root`, the user `ftp` plays a special role. If this user exists, a user can log in to the computer with the `ftp` command without needing to have an account. The user simply assumes the identity of user `ftp` or `anonymous`, and the system prompts for the user's e-mail-address as password. The home directory of user

ftp is then used as the root directory, so that a user who gains access in this way only has access to certain files.

subdirectories

However, this requires the presence of the subdirectories `dev`, `bin` and `usr` with their corresponding files in the home directory of user ftp. The exact structure of this file tree is explained on the on-line reference manual page for `ftpd`.

7.6 Shells

/etc/shells

To allow users to change their shells themselves, the file `/etc/shells` must list every shell with its path. This is often forgotten when a shell is installed afterwards. We present an example of the file /etc/shells:

```
/bin/sh
/bin/bash
/bin/ksh
/bin/tcsh
```

If there is no entry for a shell in the file `/etc/shells`, the shell cannot be used by users as a login shell and there are problems with various TCP/IP programs. Here we have the

chsh

invocation of the command `chsh` to change the login shell.

```
dirk1:/home/stefan# chsh
Changing the login shell for stefan
Enter the new value, or press return for the default

        Login shell [/bin/sh]: /bin/bash
dirk1:/home/stefan#
```

The command `chsh` permits users to change their login shell. The administrator should also use this command in the configuration of an account. In this case the respective user name must be passed as parameter.

7.7 User information

login messages

The message that is displayed before the login prompt appears is located in the file `/etc/issue`. This file normally contains

entries with a greeting message, the name of the computer, and instructions for users.

When a user is logged in, the message from the file `/etc/motd` is displayed. For systems that use shadow passwords, this can be configured in the file `/etc/login.defs`. `motd` is an acronym for "message of the day". This file should also be used as such. Some distributions, including Slackware, overwrite the files `/etc/issue` and `/etc/motd` in the start scripts of the system. To allow the user to customize these files, the commands that overwrite them must be removed from the scripts. In the Slackware distribution this occurs in `/etc/rc.d/rc.S`.

/etc/motd

7.8 Backups

The `tar` command affords a relatively simple way to make backups of important files. `tar` is one of the standard UNIX commands available under Linux in its enhanced form from the FSF.

tar

For example, to back up all data in the directory `/home/stefan` onto diskettes, the `tar` command can be invoked with option `M` (multivolume mode). When one diskette is full, `tar` prompts for the next.

diskettes

Here we have the creation of a tar archive on diskette:

```
dirk1:/root# cd /home/stefan
dirk1:/home/stefan# tar cvfM /dev/fd0 *
```

Subdirectories are automatically included in this archiving. To restore such a backup onto the hard disk, the `tar` command is invoked again with option M; otherwise it would terminate after the first diskette. This example restores a tar archive from diskettes:

```
dirk1:/root# cd /home/stefan
dirk1:/home/stefan# tar xvfM /dev/fd0
```

The M option is not available on other UNIX systems that do not use the GNU `tar` command.

Larger backups can be made onto a streamer in the same manner. Instead of `/dev/fd0`, the streamer is specified as the device.

7.9 File system management

The system administrator is also responsible for the management of the file system. In normal operation this is restricted to checking free memory at regular intervals and from time to time deleting the contents of the `/tmp` directories.

In the event of a system crash, however, the administrator must carry out a consistency check of the file system. For this purpose the individual Linux file systems provide a special tool named `fsck` (file system check). The administrator must assure that the file system to be checked is not mounted and that the appropriate check program is used. For the currently most popular `ext2` file system, the corresponding tool is `e2fsck`. The following example demonstrates a consistency check for a file system:

file system check

```
linx1:/>umount /dev/hda3
linx1:/>e2fsck /dev/hda3
linx1:/>mount /dev/hda3 /home
```

`fsck` is invoked with the partition or file system as parameter. Normally the program displays any inconsistencies on the console. If no error messages appear, the file system has not been corrupted.

Special options permit automatic error correction. However, `fsck` program authors usually recommend repairing a damaged file system interactively instead.

interactive repair

Checking a file system after it was improperly shut down due to a system crash runs automatically in the `ext2` system with the next system startup.

7.10 Updates

Because, particularly with Linux, new versions of the kernel, the C library or the C compiler are released almost every month, one of the tasks of the system administrator is to install these updates in the system. This section explains how to incorporate such updates.

GCC

GNU C compiler

The newest version of the GNU C compiler (GCC) for Linux can be downloaded from the ftp server `tsx-11.mit.edu`. It usually encompasses several files compressed with `tar` and `gzip`, and containing compiled programs (binaries) for Linux. To replace the old C compiler with the new version, it usually suffices to unpack the `tar` files in the root directory. The older version is overwritten in the process. To save storage space, the directories of the old version under `/usr/lib/gcc-lib` can be deleted. More detailed instructions for updating are included in the corresponding `README` or `release` files found in the same `tar` archive.

Libraries

Linux C library

The Linux C library is likewise downloadable along with the C compiler from the ftp server `tsx-11.mit.edu` in the directory `/pub/linux/packages/GCC`. Installation of a new version of the library usually requires two `tar` archives whose names contain the strings "image" and "inc", respectively. One file contains the actual library files, the other the matching header files. In addition there is a file with `extra` in its name that contains the libraries for debugging and profiling.

These archives must be unpacked in the root directory such that the include files are in the directory `/usr/include` and the library files in `/lib` and `/usr/lib`.

```
stef1:/# tar xvfz /home/strobel/image-4.5.26.tar.gz
./lib/
./lib/libc.so.4.5.26
./lib/libm.so.4.5.26
./lib/libc-lite.so.4.5.26
./usr/lib/
./usr/lib/libc.sa
./usr/lib/libcurses.sa
./usr/lib/libtermcap.sa
./usr/lib/libdbm.sa
./usr/lib/crt0.o
./usr/lib/libc.a
./usr/lib/libm.sa
./usr/lib/libm.a
./usr/lib/libtermcap.a
./usr/lib/libcurses.a
./usr/lib/libdbm.a
./usr/lib/libbsd.a
./usr/lib/libieee.a
stef1:/# tar xvfz /home/strobel/extra-4.5.26.tar.gz
./usr/lib/
./usr/lib/libg.a
./usr/lib/libmcheck.a
./usr/lib/gcrt0.o
./usr/lib/libc_p.a
./usr/lib/libgmon.a
stef1:/# tar xvfz /home/strobel/inc-4.5.26.tar.gz
./usr/include/
./usr/include/arpa/
./usr/include/arpa/ftp.h
./usr/include/arpa/inet.h
...
```

major/minor version number

The version number of the libraries consists of two parts, a major and a minor version number. The files that are unpacked above to `/lib` with major version 4 and minor version 5.26, for example, would be called `/lib/libc.so.4.5.26` and `/lib/libm.so.4.5.26`.

symbolic link

These files are accessed via symbolic links that only contain the major version in their names. For the above files there must be symbolic links to `/lib/libc.so.4` and `/lib/libm.so.4` that reference the actual files. When a library with a new minor version is installed, these links need to be modified. Here we demonstrate the installation of a new library with the invocation of the program `ldconfig`.

```
stef1:/# /sbin/ldconfig -nv /lib
/sbin/ldconfig: version 1.4.3
/lib:
        libc-lite.so.4 => libc-lite.so.4.5.26
        libgr.so.1 => libgr.so.1.3
        libXpm.so.3 => libXpm.so.3.3.0
        libvga.so.1 => libvga.so.1.0.11
        libm.so.4 => libm.so.4.5.26
        libc.so.4 => libc.so.4.5.26
stef1:/#
stef1:/# cd lib
stef1:/lib# ls -l libc* libm*
lrwxrwxrwx  1 root root      19 Apr 27 23:50 libc-lite.so.4 -> libc-lite.so.4.5.26*
-rwxr-xr-x  1 root root 619524 Apr  4 21:32 libc-lite.so.4.5.26*
lrwxrwxrwx  1 root root      14 Apr 27 23:50 libc.so.4 -> libc.so.4.5.26*
-rwxr-xr-x  1 root root  23620 Apr 19 14:05 libc.so.4.5.24*
-rwxr-xr-x  1 root root 623620 Apr  4 21:31 libc.so.4.5.26*
lrwxrwxrwx  1 root root      14 Apr 27 23:50 libm.so.4 -> libm.so.4.5.26*
-rwxr-xr-x  1 root root 107524 Apr 19 14:05 libm.so.4.5.24*
-rwxr-xr-x  1 root root 107524 Apr  4 21:31 libm.so.4.5.26*
stef1:/lib#
```

Kernel

The source code of the newest kernel can be downloaded from most Linux ftp servers. However, it first appears on Finland's `nic.funet.fi`. To install the newest version of the kernel, we recommend first completely deleting `/usr/src/linux` and then unpacking the `tar` archive with the kernel source code into the directory `/usr/src`. In the process the subdirectory `linux` is created. As described in Section 6.2 on the configuration and compilation of the kernel, the drivers to be used can be defined with `make config` and then the kernel can be compiled.

kernel source

7.11 Boot diskette

If the administrator has forgotten the root password, or if an error occurred during the installation of a new C library, and so the administrator can no longer log in, then a boot diskette is usually the only salvation. A simple boot diskette contains only the Linux kernel, which was written to a diskette with the utility `dd`. The file system that this kernel mounts as root file system can be specified with the program `rdev`.

If the root file system is corrupted or programs necessary for booting on this file system are no longer executable, then a normal boot diskette will not suffice. This calls for a boot diskette that also contains its own root file system. SLS installation diskette A1 or the Slackware boot and root diskette can be used

root file system

for this purpose. After the system has been booted from the diskette, the root file system of the hard disk can be mounted to allow correction of the defective files.

Support and Help

One argument that is frequently posed, especially in commercial settings, to reject free or public domain software is the lack of a licensing company that provides support and a hotline. In the USA some companies have recognized this market niche and offer commercial support for free software. A similar trend is developing in Europe as well.

freeware support

Apart from such support, there are many possibilities to receive information and help via the Internet or directly from the Linux system for concrete problems. This chapter offers an overview of these possibilities and of the available documentation for Linux.

8.1 man, xman

As with every UNIX system, a simple source of information is the on-line documentation. Each of these files describes, for example, one command, one C library routine, a device, or the contents of a configuration file. The command for displaying these on-line reference manual pages is `man`.

on-line reference manual

The on-line reference manual pages are divided into sections from 1 to 8 and are located under the directory `/usr/man/` in the subdirectories `man1` to `man8` and `cat1` to `cat8`. In addition, there are the directories `mann` and `catn`, which contain new on-line reference manual pages. Here we have the directories of the on-line reference manual pages:

```
linux2:/usr/man#ls
./      cat1/   cat3/   cat5/   cat7/   catn/   man3/   man6/   mann/
../     cat2/   cat4/   cat6/   cat8/   man1/   man5/   man8/   whatis
linux2:/usr/man#ls man1
./          cpp.1       othello.1   uupath.1    xmag.man    xtetris.1
../         g++.1       pathto.1    uuwho.1     xmake5.1
cccp.1      gcc.1       spider.1    wish.1      xpuzzle.1
linux2:/usr/man#
```

The command man finds the on-line reference manual page files independently. The user need not first change to a subdirectory of /usr/man to use the command. The following example shows the invocation of man to display the on-line reference manual page for the command mv.

```
linux2:/root# man mv
```

Format of the on-line reference manual pages

nroff

The on-line reference manual pages are sometimes available in two or three different formats. The original format is usually source code for nroff and is located in the directories man1 to man8. This format is unsuitable for direct screen output, and must be translated by the text formatting program nroff or groff (Also refer to Section 11.4).

readable form

compression

For more complex on-line reference manual pages this procedure takes some time; therefore, after the files have been translated once, they are stored in readable form in the directories cat1 to cat8. To do this, however, the user must have write permissions for these directories. If desired, the files can also be compressed with compress, which only insignificantly increases the time required for displaying them, but radically reduces disk storage consumption.

manual path

If additional on-line reference manual pages have been installed in a directory other than /usr/man (for example, /usr/local/man), the path for the on-line reference manual pages can be defined with the environment variable MANPATH. The following lines define the path for the on-line reference manual pages in a Bourne shell:

```
linux2:/root# MANPATH=/usr/man:/usr/openwin/man:/usr/local/man
linux2:/root# export MANPATH
```

This example defines the path for the on-line reference manual pages in a C shell:

```
linux2:/root# setenv MANPATH /usr/man:/usr/openwin/man:/usr/local/man
```

xman

The X Window System affords the somewhat more comfortable utility `xman`.

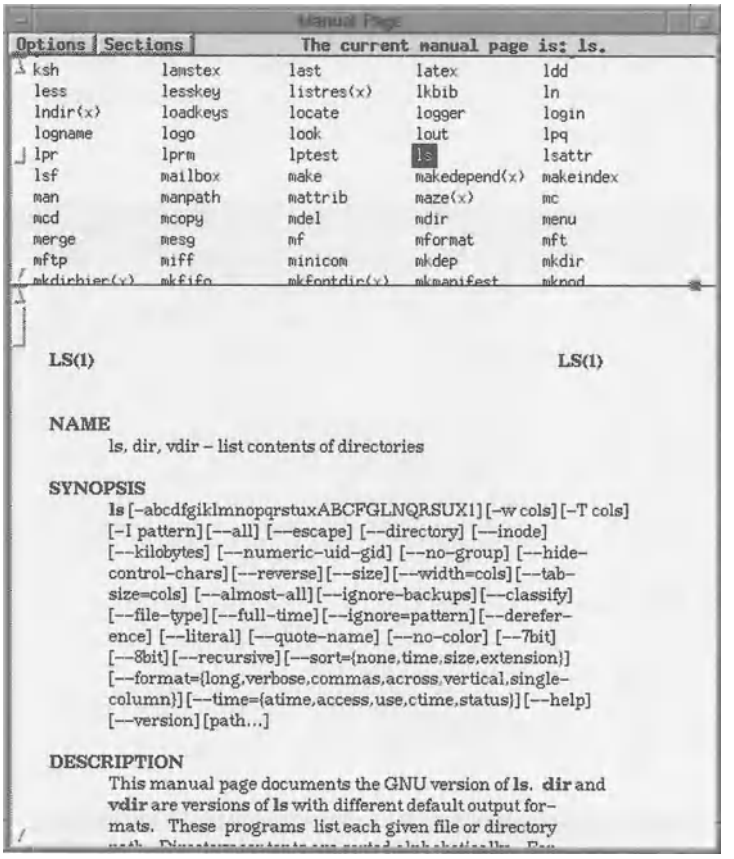


Fig. 8.1 Graphical display of reference manual pages with xman

Here the user first selects a section of the manual. Then the utility provides an overview of all on-line reference manual pages in that section. With the mouse the user can select an on-line reference manual page from this overview for display.

When a problem arises with a command or with the configuration of a program, the first point of reference should always be the relevant page of the on-line reference manual. Naturally the `man` command also has its corresponding on-line reference manual page, which the user invokes with `man man`.

8.2 Info

emacs

The documentation of many FSF programs is available in GNU info format. GNU Emacs uses this format to provide hypertext-like navigation through these documents. Within the Emacs editor the user invokes the info mode and selects the needed text from the menu of available info documents.

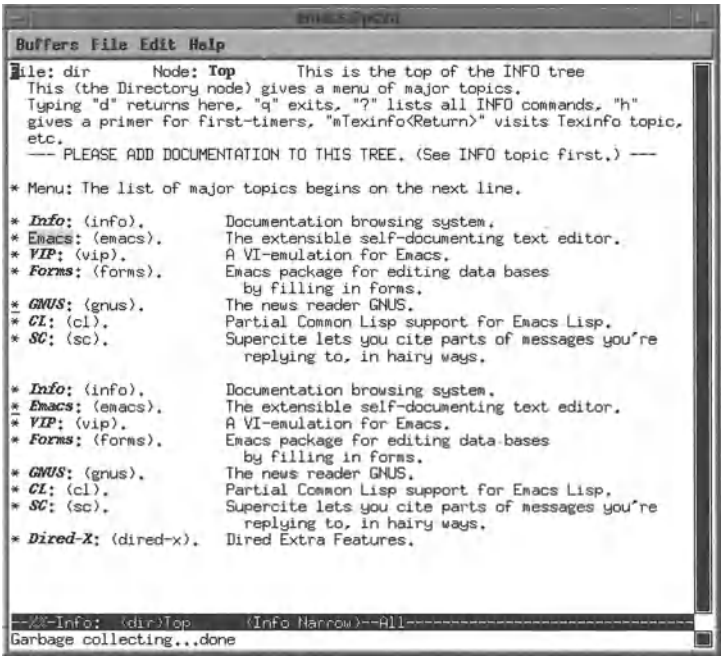


Fig. 8.2 GNU info pages in Emacs

In an info file the user can navigate hierarchically from keyword to keyword (see Chapter 11). Newer versions of the Emacs editor, such as Lucid Emacs and Emacs 19, afford the possibility to navigate with the mouse under the X Window System. There are also programs like `xinfo` and the Tcl/Tk-based `tkinfo` that were conceived exclusively for displaying info files.

Examples of info documents include the documentation to the GNU C compiler and the GNU AWK utility.

8.3 Newsgroups

Since Linux continues to be developed on the Internet, the many communication services of the Internet are utilized to the fullest. Among these services, the newsgroups provide an information source of particular interest to Linux novices.

Linux enjoys several newsgroups that provide a forum for discussion of questions about the system and its installation as well as other subjects. The most important newsgroups are:

- `comp.os.linux.announce` (affectionately c.o.l.a)
- `comp.os.linux.help`
- `comp.os.linux.misc`
- `comp.os.linux.admin`
- `comp.os.linux.development`

`comp.os.linux.announce` is a moderated newsgroup for the announcement of new programs or ports to Linux. Messages intended for posting in a moderated group are not posted directly, but are first filtered by a moderator who is responsible for upholding the rules within the newsgroup.

announcements

The other four groups have no restrictions. Any participant can post messages or questions here. However, users should be aware that every message posted to a newsgroup is propagated around the world and consumes storage on every news server.

Currently over 100 new postings appear in these newsgroups daily. This makes it difficult to keep up to date without perusing these postings more than on a weekly basis. A newsgroup

observer can often solve problems simply by reading the newsgroup postings and following the problems and solutions that other users share.

8.4 FAQs and HOWTOs

questions

FAQs (frequently asked questions) provide another important information source that is closely linked to the newsgroups. A FAQ is a list of questions that arise often in postings and, more important, the answers to these questions.

FAQs cover a broad range of subjects. They are usually compiled by active participants of the newsgroup, with the intention of avoiding the repeated posting of the same questions. A FAQ thus consists of a series of such questions and the corresponding responses provided by various competent newsgroup participants.

answers

Some of the original Linux FAQs have evolved into HOWTOs. These documents resemble FAQs, but contain more detailed text.

Both FAQs and HOWTOs are plain text files, so that they can be read and printed with any text editor, with the UNIX `more` command, or even with DOS.

UNIX FAQs

As a rule, Linux FAQs and HOWTOs do not cover general UNIX questions. A dedicated newsgroup exists for this purpose (`comp.unix.questions`) with its own FAQs. Thus UNIX novices might also want to browse the UNIX FAQs.

Sources for FAQs

news.answers

The best source of FAQs of all kinds is a special newsgroup entitled `news.answers`. Many groups regularly post their FAQs here. In addition to FAQs on Linux, UNIX and programming languages, a new explorer can find FAQs on other subjects of interest outside the realm of computer science.

The special Linux FAQs are posted regularly in the newsgroup `comp.os.linux.announce`. Naturally, current FAQs can also be found on the many ftp servers, usually in a

subdirectory `doc` under `Linux`. For example, the ftp server `nic.funet.fi` stores current Linux FAQs in the directory `/pub/OS/Linux/doc/FAQ`.

8.5 Mailing lists

Various mailing lists have been created for kernel hackers and other active Linux system collaborators. These mailing lists primarily support the exchange of news of new developments, problems, ideas, and patches. Messages sent to the address of the mailing list are collected and forwarded several times a day to all addressees on the mailing list.

channel

To join the mailing list for a certain subject, called a *channel*, the user sends a message to the address `linux-activists-request@niksula.hut.fi`; the reply will contain detailed instructions for using the mailing list. In the simplest case the following command suffices to request help for mailing lists:

```
echo help | mail linux-activists-request@niksula.hut.fi
```

newbie

Since these mailing lists are intended primarily for Linux developers, a novice should not pose questions through the normal channels. Novices have their own *newbie* channel.

8.6 Other documents

Linux Documentation Project

The Linux Documentation Project (LDP) is currently working to produce detailed manuals for Linux. Beyond the *Linux User's Guide* and the *Linux Installation Guide*, this documentation also includes more complex documents such as the *Network Administration Guide* (NAG), which covers almost all aspects of network installation under Linux.

kernel

The *Kernel Hacker's Guide* (KHG) deals with the structure of the kernel and the development of device drivers for Linux. It also provides an excellent glimpse into the inner workings of Linux.

Although some of these manuals are incomplete, they can be procured via ftp and printed out. Some bookstores sell printed and bound copies of these manuals. The public-domain files can be drawn from the ftp server `sunsite.unc.edu` in the directory `/pub/Linux/docs/LDP`.

8.7 Other sources

Besides the sources identified above, there are many other Internet services and organizations that provide information and documentation. This section identifies some of them.

WAIS

WAIS server

WAIS (Wide Area Information System) is an Internet service that supports searching WAIS servers worldwide by keyword for documents of all kinds. WAIS also affords a possibility for obtaining manuals and FAQs. A list of WAIS servers and a detailed description of WAIS can be drawn by ftp from the host `think.com` or from the `comp.infosystems.wais` newsgroup.

README files

instructions

As with most larger programs, the Linux kernel documentation is complemented by release notes and `README` files containing important instructions for its installation and operation. These files normally reside in the same directory as the system or the source code of the system.

For example, the directory `/usr/src` contains many subdirectories with system components and utilities. Almost all of these subdirectories contain their own `README` files. Hence in the directory `/usr/src/lilo` we usually find the Linux Loader (LILO) along with a `README` file that describes both installation and configuration in detail.

X Window System

As the propagation of graphical workstations began, there were scarcely any standards for programming graphical user interfaces (GUIs). Most manufacturers provided their own GUIs with their machines.

GUIs

If an application that was intended to exploit the graphical possibilities of the new machines was to run on various platforms, it had to be developed and maintained in multiple variants. Large institutions that worked with systems from several manufacturers felt the brunt of these problems.

This prompted the Athena Project at the Massachusetts Institute of Technology (MIT) and DEC to develop a platform-independent, uniform environment for the development of graphical applications. At first the development of the X Window System received financial support only from DEC and IBM.

Athena Project

In January 1988 twelve renowned workstation producers joined to form the X Consortium. This institution's goal was to promote the further development and standardization of the X Window System and to enable commercial utilization.

X Consortium

In the same year the X Window System Version 11 (X11 R1) appeared. Contrary to previous versions, Version 11 had outgrown the research stage. Although the new release was no longer compatible with Version 10, it afforded much greater flexibility and performance, which made it suitable for commercial application.

version 11

9.1 Features

The X Window System boasts features that distinguish it from conventional graphical user interfaces such as the Apple Finder and MS-Windows. The following subsections describe the most important concepts of this powerful system.

Openness

Contrary to most other GUIs, the X Window System was conceived from the start as an open system. This means that the developers maintained independence from any manufacturer-specific policy and that the complete source code is available for free.

The X Window System supports comfortable development of portable and hardware-independent software. The programmer need not worry about the hardware platform. The system supports numerous input and output devices. Interfaces are provided for manufacturer-specific extensions, which allows the connection of specialized hardware.

X11 acceptance

The manufacturer independence and the extreme portability of the X Window System have won it a high level of acceptance on the workstation sector. Today there is hardly a hardware platform, from PC to mainframe, for which the system is not available.

XFree86 server

This general availability also proves to be an advantage for Linux users. The X Window System server XFree86 that runs under Linux displays extremely high performance on normal PC hardware and is in any case equal to commercial X servers, and in some areas even proves superior.

Client/server architecture

Due to its internal structure, the X Window System distinguishes between the X server and the X clients. The server program is responsible for the management of local hardware such as monitor, keyboard, and mouse, and provides the interface

between the user and the individual X applications, which are the X clients.

In general a workstation runs only one server, which provides any number of clients with input and handles screen output requested by these clients. However, it is quite possible that one server might manage multiple monitors that are connected to a single workstation, which is particularly interesting for CAD systems.

X protocol

The X protocol makes the only connection between the server and the X clients. Due to this standard protocol the clients could even be running on other computers in the network (see "Network transparency" below).

client/server

The reader needs to be clear that the designations X server and X client deviate from the accustomed terminology. Normally a server provides the back end (e.g., in SQL databases) and a client provides the front end. Under the X Window System this is usually inverted: the server runs the front end, while the client runs the back end. Often this distinction disappears because the server and the client are running on the same machine.

This distinction between client and server has some consequences for both the programmer and the user. For example, the programmer has to transfer a graphic that the client has in bitmap form over the network to the server before it can be displayed. These interrelationships need to be considered in the development of X11 applications to avoid creating unnecessary network traffic and thus provoking the resulting loss in execution speed.

network traffic

response time

Since an X client is relatively loosely coupled to the server, the inexperienced user can encounter some surprises from time to time. For example, if a client does not respond immediately to user input because of high network or CPU load, many users tend to repeat the input. However, since the X server records every event and definitely sends it to the client, the action could be executed repeatedly, which only very seldom satisfies the intentions of the user.

Network transparency

An important feature of the X Window System is its network transparency. Most workstations on the UNIX sector are connected via network to one another. The X Window System provides a mechanism for redirecting the output of an application to an arbitrary workstation on the network.

This feature permits running computation-intensive programs on the most powerful CPU while the input and output are handled by a smaller workstation on the network. If an appropriately fast network connection is available, the two machines could even be located thousands of miles apart.

protocol

Although it currently supports only TCP/IP and DECnet, the X Window System is principally not bound to a specific network protocol.

performance

Surprisingly, this network transparency hardly leads to a loss in execution speed compared to conventional graphic systems. The X Window System also supports the display of local applications as well as programs running on different machines on the network all on one monitor. Therefore the X Window System provides an ideal basis for integration in all possible application domains.

9.2 Structure

The structure of the X Window System reflects multiple levels. Although the user normally does not perceive these levels, understanding this structure should help to clarify some of the phenomena that the X Window System user experiences.

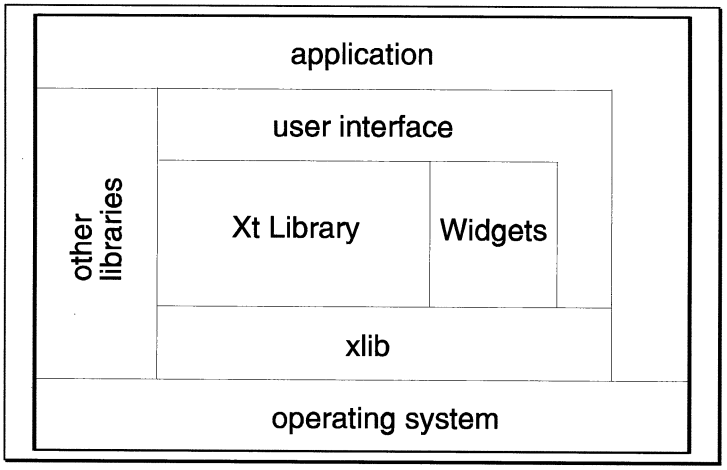


Fig. 9.1 Structure of an X application

Xlib and X-protocol

Xlib interface

A very extensive graphic library with a standardized C interface named Xlib provides the basis of the X Window System. Every invocation of an Xlib routine is translated to a corresponding data stream that is then transferred via the network (X protocol level) and can be interpreted on another workstation. This also guarantees problem-free communication between workstations of different manufacturers.

X11 does not permit any kind of direct access to the video hardware and does not provide any means for circumventing its standardized interface.

Intrinsics

X Toolkit

Since Xlib permits only rudimentary graphic operations such as drawing lines and circles and filling surfaces, higher levels have been introduced. With the X Toolkit, MIT made a highly mature proposal for the implementation of such libraries. For creating graphical user interfaces, the developer normally uses objects on the toolkit level. If an application needs additional primitive

graphical output operations, these are realized with direct Xlib invocations.

Xt library

The X Toolkit Intrinsics library (Xt library) supports the development and use of complex graphical objects such as buttons, text input fields, and selection menus. These objects are collectively called widgets.

Widgets

The actual look and feel of such objects is not determined further by the intrinsics library. A design goal in the development of the X Window System was not to prescribe the look of applications that build on it, but only to provide interfaces for the implementation and use of such widget sets. Thus over time various sources produced such widget libraries, sometimes with significant deviations in look and feel. However, this multiplicity seems to have tended to confuse users. Recent developments show that

Motif

most manufacturers have agreed on the Motif Widget Set of the Open Software Foundation as the de facto standard. This suggests that the future will likely see a uniform concept for graphical user interfaces under X11.

9.3 Resources

object-oriented

In their design the developers of the X Window System created an unusually flexible interface for the configuration of various parameters of the system. The basis was the object-oriented approach of the X Toolkit and the widget sets building on it.

widget attributes

Each widget has a number of specific attributes such as position, size, shape and color. These attributes can be influenced by the programmer as well as later by the user.

default settings

Every graphical object has internal default settings as determined by the widget developers. The programmer of an X application normally modifies these settings only as much as necessary for the specific requirements.

resource manager

When an X application starts, the resource manager loads any new data from the X resource database, whose contents the user

can modify in several stages. First, most X clients are provided with the appropriate application default file, which is copied into the X Window System directories. This file contains base settings (such as size and position of the graphical objects and error message texts in the respective national language) that are important for the application. In addition, each user's home directory can contain a customized `.Xdefaults` file to retain the user's preferences.

The ASCII format representation of these resource settings in the file can seem quite complicated for the inexperienced user. To distinguish it within a resource database, each application is assigned a name (class) by its programmer that does not need to match the name of the program file (instance). Likewise each widget that can be configured from the outside has such a designation. To reference a widget unambiguously, its name alone does not suffice. Instead, as for a file system, a path must be specified that represents the widget hierarchy. To allow simultaneous manipulation of the attributes of multiple widgets, wildcards are permitted within such a path.

widget name

widget path

Here we have an excerpt from a resource database:

```
XTerm*ScrollBar:            true
XTerm*Foreground:           white
XTerm*Background:           gray20
XTerm*IconName:             XTerm
XTerm*WaitForMap:           true
XTerm*MarginBell:           false
XTerm*JumpScroll:           true

Editor.view.background:     gray
Editor.close.labelString:   Close
Editor.open.labelString:    Open
Editor*background:          gray90
```

The first column of a resource file specifies the attribute to be manipulated. This usually corresponds to a widget resource. However, the programmer can define new, application-specific resources. The hierarchy and the names of the available resources of a program can usually be found in the corresponding on-line reference manual page.

Version 11 Release 5 of the X Window System includes an interactive resource editor (`editres`) that permits comfortable manipulation of all resource settings of a running program and stores these in an ASCII file if the user desires. It is noteworthy

resource editor

that this is possible at run time of a program. This allows the user to gain an immediate impression of the effects of modifications. Unfortunately, the protocol required by `editres` is not supported yet by all widget sets, which prevents the universal application of the tool.

9.4 Window manager

The window manager is a special client that usually exists only once per workstation. Its job is managing the various windows of a display to allow the user to modify the position and size of windows. The window manager also handles the window decoration, which designates the peripheral regions of a window and its various control elements. Naturally the individual X client is responsible for the contents of a window.

window decoration

The X Window System likewise does not prescribe the look and feel for the window manager. To assure problem-free communication between a given window manager and the clients running under its control, the X Consortium passed the ICCC (Inter-Client Communications Conventions).

ICCC

Users have a broad choice of different window managers, each having its advantages and drawbacks. The standard X distribution, for example, contains the `twm` (Tab Window Manager), which affords little in the way of comfortable operation. The Open Software Foundation (OSF) created the Motif window manager (`mwm`), whose look and feel matches the Motif widget set.

Motif

Sun Microsystems' OpenLook window manager (`olwm`) is available free as source code. `olwm` was extended by a third party to the OpenLook virtual window manager (`olvwm`), which provides almost any number of virtual screens and allows switching between them with the desktop manager.

olvwm

However, most Linux distributions favor `fvwm` by Robert Nation. This window manager is also based on `twm`, but consumes significantly less memory than the original. By way of a configuration file (`.fvwmrc`) and by setting numerous parameters, the user can exert significant influence on the look and feel of `fvwm`. This allows the user to mimic the look and feel

fvwm

of mwm or the window manager by Silicon Graphics. A virtual desktop supports multiple screens, similar to olvwm. Especially noteworthy is the feature allowing the linking of external modules, which can then communicate with the window manager via a defined interface. One such module is GoodStuff, an icon bar on which the user can place icons for the most frequently used applications on the desktop and from which they can be launched. Another module permits the coupling of sounds to certain user actions such as the opening and closing of a window.

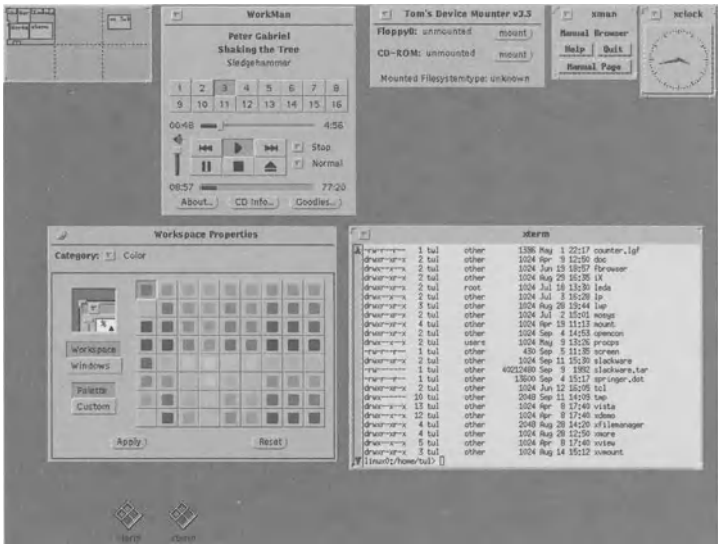


Fig. 9.2 OpenLook Virtual Window Manager (olvwm)

Another option is the generic window manager (gwm), whose look and feel the user can adapt to either of the two standards (OpenLook and Motif). ctwm is an offspring of twm, but offers a more pleasing look and feel and is more configurable than its predecessor.

gwm

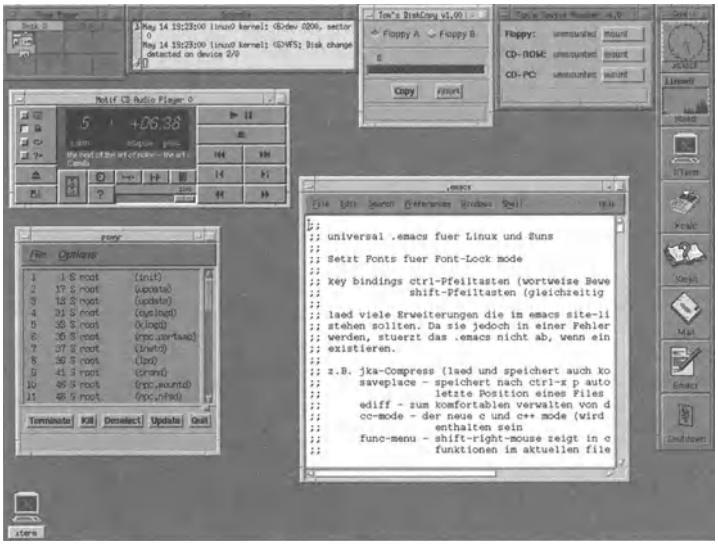


Fig. 9.3 Window manager fvwm

9.5 Toolkits

This section introduces the most important libraries (toolkits) for the production of graphical user interfaces under the X Window System.

Athena widget set

MIT

The first available widget set for the X Window System was the Athena widget set, which remains its standard that is included in the MIT distribution. Many early X applications use the Athena widgets. Even today many freeware programs continue to use these widgets.

Unfortunately, the somewhat antiquated look and feel of some of its elements has led manufacturers of commercial applications to adopt more modern toolkits. Due to the similarities between intrinsics-based toolkits, this certainly posed no great problem.

However, the look and feel of the Athena widgets has recently received a face lifting. The free availability of Athena's source

code made it possible to exchange drawing routines. The modifications led to a Motif-like three-dimensional look. Interestingly, in Linux the new library (`libXaw3D`) can simply replace the old one (`libXaw`).

OpenLook

OpenLook itself represents only a detailed specification of the look and feel for graphical user interfaces. After several years of development, it was released by AT&T and Sun Microsystems with the goal of setting a standard for graphical user interfaces under X11.

On the basis of the OpenLook specification, Sun developed a toolkit for X Windows (XView) that is nearly compatible with its previous graphic system SunView. This toolkit was included in Release 4 of the MIT distribution of X11, making it public domain. Unfortunately XView builds directly on Xlib, which means that it is not a "true" widget set.

XView

Sun used this toolkit to implement a series of standard applications that are delivered with every Sun workstation (OpenWindows Deskset). By delivering OpenLook with its UNIX System V systems, AT&T contributed to OpenLook's wide propagation. As an alternative to XView, an X Toolkit-based widget set that likewise met the OpenLook specifications was developed primarily by AT&T.

OSF/Motif

To spur the development of new technologies and standards, especially for UNIX systems, most of the renowned UNIX manufacturers (including Hewlett Packard, IBM, and DEC, but not Sun or AT&T) joined forces in the Open Software Foundation (OSF).

Open Software Foundation

This group also recognized the need for a uniform user interface for X11 and initiated the development of OSF/Motif. DEC and HP were involved substantially in the design and implementation. Within a few months a user interface emerged

that had been adapted to the look of common GUIs on the PC sector. Its main components were the Motif window manger (mwm) and the X intrinsics-based Motif widget set.

User Interface Language

In addition, the formal User Interface Language (UIL) was defined that enabled the simple description of Motif-based graphical user interfaces. This description can be translated to a binary format with a special compiler and interpreted with the help of a library.

Numerous third parties offer programs for the interactive creation of Motif graphical user interfaces.

Due to the large number of companies involved in OSF, Motif soon became the de facto standard of the UNIX world. OpenLook developers Sun Microsystems and AT&T (and later UNIX System Laboratories and Novell) merged with the Motif interface trend in the spring of 1993 with the COSE Initiative (Common Open Software Environment). This delivered the coup de grâce for OpenLook, at least on the commercial sector.

COSE

Since OSF did not make the source code of Motif available for free, users have to pay fees for each run-time license. For this reason Motif was not available for Linux from the start. Meanwhile there are commercial Linux versions of OSF/Motif at reasonable prices. Universities can purchase the source code directly from the OSF and compile it on their workstations, which was done at various sites for Linux systems.

Motif for Linux

SUIT

SUIT (Simple User Interface Toolkit) was developed at the University of Virginia. Contrary to most X toolkits, this toolkit is also available for other common user interfaces such as MS-Windows and Apple Finder. SUIT enables the development of applications that are portable between these operating systems. Its look and feel strongly resembles OSF/Motif, which should help its user acceptance.

portability

Motif look

A particular feature of this library is its integrated interface editor, which enables the user to manipulate the interface easily and interactively. For example, the user can modify the position

and form of the objects without needing to recompile the application. When the user quits the application, all modifications are automatically written to a file.

One drawback is that the code of the editor, which is linked to each program, significantly increases storage requirements if it is linked statically. A shared library version ameliorates this problem. However, the sluggish response to user input also proves to be a flaw.

InterViews

InterViews is an extensive graphical environment that was developed at Stanford University. In contrast to X Toolkit and its widget sets, InterViews has a C++ interface and thus incorporates object-oriented concepts. In addition to its graphic library, the InterViews system provides an interactive interface editor (`ibuild`), a WYSIWYG text editor (`doc`), a C++ class browser (`iclass`), and a powerful, vector-oriented drawing program (`idraw`).

C++

Due to the quite substantial overhead of most object-oriented systems, InterViews also puts significant demands on the hardware.

Tcl/Tk

Tcl/Tk, another development of the University of California at Berkeley, permits the writing of scripts with graphical front ends. Tcl (pronounced tickle, for Tool Command Language) itself has nothing to do with graphical user interfaces. Instead, Tcl in an interpreter language that is available as a C library. This makes it easy to link to existing C programs. Tcl includes powerful constructs for processing lists and associative arrays, and its language scope can be extended via a C interface.

front ends

Tk is a widget set that provides both a C interface and a Tcl interface. This makes Tk particularly suitable for providing Tcl programs with a graphical user interface. Because of the underlying interpreter language, prototypes as well as complete

prototyping

applications can be developed easily and quickly. Prominent examples are Picasso, tkman, Zircon, and tkined.

XF

The Tcl/Tk program XF, supports the graphical, interactive creation of a Tk interface. This program allows not only the simple realization of Tk interfaces but also the loading and later modification or extension of existing Tcl/Tk programs. Since XF itself was developed in Tcl/Tk, the interface can be tested in the course of development and corresponds exactly to that of the finished program.

We present additional details about Tcl in Section 10.5.

Tcl/Motif

Motif binding

Tcl/Motif combines the advantages of Tcl with those of the Motif Widget Set. This package uses normal Motif widgets for the creation of graphical user interfaces. Hence Motif interfaces can easily be constructed and tested in an interpreter environment. All the accustomed widget resources are at the programmers disposal. Tcl/Motif even supports the drag & drop mechanism of Motif 1.2.

XView/Slingshot/UIT

XView

XView is an independent toolkit that is not based on Xt intrinsics. The Sun Microsystems product was delivered with Release 4 of the X Window System. The programming interface depends

SunView

strongly on the antiquated SunView, the non-network-transparent, highly kernel-dependent graphical user interface on older Sun workstations. XView implemented the look and feel of Open-Look and, because of the object-oriented approach of its not overly complex structure, was relatively easy to program.

To extend the features of XView, a Sun Microsystems employee in England developed the Slingshot extensions. This package offers numerous new graphical objects that harmon-

Slingshot

iously integrate into the base system. Slingshot extends the object-oriented approach with rudimentary graphical figures such as lines and rectangles. The Slingshot extensions also simplify the

programming of the drag-and-drop functionality that was already relatively mature in OpenLook.

To adapt XView (and Slingshot) to the newest developments in the field of programming languages, another Sun employee developed a C++ class hierarchy (UIC), which permits the utilization of all XView objects and simplifies using them. Unfortunately, the integration of functions in derived C++ classes can pose some problems.

UIC

9.6 Interface builder

The most comfortable way to produce X Window-based applications is by using specialized programs for the interactive creation of graphical user interfaces. Such interface builders usually generate C source code, which the programmer can extend as needed.

Several tools have become available for the interactive design of graphical user interfaces under Linux. ParcPlace's ObjectBuilder, an extremely powerful tool for interface development in C++, provides an Xlib-based C++ class hierarchy whose look and feel can be switched at run time between OpenLook and Motif. Only the Linux version of this powerful tool is freeware and is included in many distributions.

ParcPlace

ObjectBuilder

Flux GbR in Berlin also offers its Motif interface builder MinD for Linux. This product's goal is to enable even the inexperienced developer to construct OSF/Motif interfaces. All graphical objects can be manipulated and positioned with the mouse. MinD then generates the C source code and a matching makefile. As is common under Motif, the functionality of the application is implemented with callback routines in C. MinD also manages the source code written by the user. Furthermore, the C compiler can be started with the click of a mouse, so that the development environment need not be left. MinD's generated source code requires no additional library routines except the Motif library and can thus easily be ported to other UNIX platforms. Since MinD is not necessarily intended for the professional programmer, not all features of Motif are supported, which impedes the development of complex interfaces.

MinD

9.7 The XFree86 server

A central component of every Linux distribution is the X Window System from the XFree86 team. XFree86 is a nonprofit organization that is primarily involved in the further development of the X server on Intel-based UNIX systems. Special attention is given to the support of up-to-date PC graphic boards. For this reason the XFree86 server also demonstrates performance that sometimes even exceeds what can be expected of RISC workstations.

performance

The standard configuration supports only common VGA graphic boards in all possible resolutions and vertical sweep frequencies. With special servers that exploit the possibilities of modern accelerator boards (ATI Mach, S3), significantly better results can be achieved.

9.8 Practice

This section first shows how a user can log in on another workstation and redirect output from X Window System programs running there to the local Linux machine. Then we briefly expound on the configuration of the X Window System from the user's point of view.

Linux as X terminal

A PC running under Linux and XFree86 is superbly suited as an X terminal. This proves especially interesting when expensive software packages are available on a workstation and should be utilized decentrally.

Numerous X servers are available that run under MS-Windows. Their performance is usually significantly below that of XFree86 servers because they translate the X protocol commands into slower MS-Windows invocations.

To give the reader a more precise picture of the possibilities afforded by using Linux as an X terminal, we present an example of such an application in a heterogeneous network.

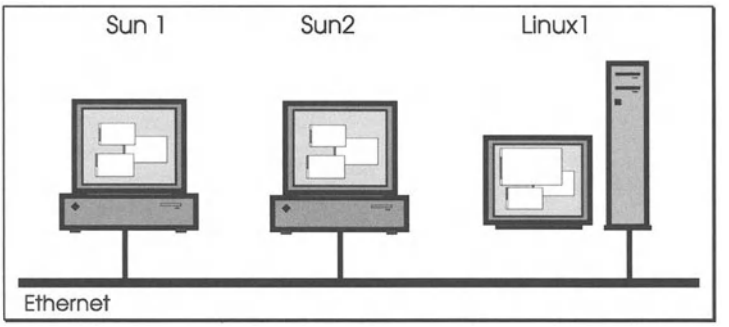


Fig. 9.4 Heterogeneous network

On a network we have a powerful graphic program available on a Sun workstation (`sun1`) that is to be used by a Linux workstation (`linux1`). The Linux PC must be connected with the workstation via Ethernet. First, external access must be permitted to the local X server with the command `xhost`. The permissions can be defined separately for each machine. In general, however, the execution of `xhost +` suffices. This terminal capture demonstrates the `xhost` command:

xhost

```
linux1:/home/stefan>xhost +
access control disabled, clients can connect from any host
xhost:  must be on local machine to enable or disable access
control.
linux1:/home/stefan>
```

Then the Linux user logs in with `telnet` or `rlogin` on the computer `sun1` :

```
linux1:/home/stefan>
linux1:/home/stefan>telnet sun1
Trying 141.7.1.20...
Connected to sun1.
Escape character is '^]'.

SunOS UNIX (sun1)

login: strobel
Password:
Last login: Mon Aug 30 09:59:51 from SunServ
SunOS Release 4.1.2 (NEWKERN) #1: Wed Dec 23 11:02:57 MET 1992
sun1:/home2/sun1/strobel>
sun1:/home2/sun1/strobel> setenv DISPLAY linux1:0.0
```

Next the DISPLAY environment variable is set so that the X library redirects all graphical output to the Linux machine. In the

setenv

C shell, as the above example shows, this is done with the command `setenv`.

If an X client is started now on `sun1`, then all windows will appear not on `sun1` but on the display of the Linux computer named `linux1`. In this way multiple applications from different workstations can be output on one display and operated from one machine.

rsh

Alternatively an X application can be started on a remote computer with the `rsh` command. This requires a corresponding entry in the `.rhosts` file on the respective machine. Section 3.3 provides more details. The following statement starts the application `xterm` on `sun1` and redirects its output to `linux1`. The option `-display` is supported by almost all X applications.

```
/home/uhl: rsh sun1 "xterm -display linux1:0.0"
```

To simplify the redirection of a remote application, it makes sense to make an entry in one of the pop-up menus of the window manager. How this functions is shown below under the configuration of `fvwm`.

Configuration of X Applications

.Xdefaults

The most important file for user-specific settings within an X Window System environment is `.Xdefaults`. This file is located in the user's home directory. The following shows the `.Xdefaults` file:

```
*fontList:  fixed

xterm*ScrollBar:        true
xterm*Foreground:       white
xterm*Background:       grey20
xterm*IconName:         XTerm
xterm*WaitForMap:       true
xterm*MarginBell:       false
xterm*JumpScroll:       true
```

resource definition

The specifications of a resource definition can refer to one or more widgets. If the definition begins with an asterisk, then it applies to all applications. In the above example the character set

for all OSF/Motif programs (only they recognize a resource named `fontList`) was set to `fixed`.

Configuration of the Window Managers

The user can configure not just the look of individual applications, but of most X window managers as well. Since many Linux users probably use `fvwm`, we go into detail only on this window manager. General parameters of `fvwm` are also modified via the file .fvwmrc in the user's home directory. If this file is not present, then fvwm looks for a file named `system.fvwmrc` in the directory `/usr/X11/lib/fvwm`.

The user should place important commands in a pop-up menu of the window manager. Such a window appears upon a click in the root window. Pop-up menus can be nested to any depth, although this does not necessarily improve clarity. Alternatively, the icons of important programs can be placed in the icon bar of the GoodStuff module of `fvwm`. We present an exemplary configuration of the `fvwm` window manager:

```
#################################################################
#
# configuration file for fvwm
#
#################################################################
#
# colors
#

StdForeColor                    Black
StdBackColor                    #d3d3d3

HiForeColor                     Black
HiBackColor                     #5f9ea0

StickyForeColor                 Black
StickyBackColor                 #908090

MenuForeColor           Black
MenuBackColor           grey
MenuStippleColor        SlateGrey

#################################################################
# fonts
Font                            fixed
WindowFont                      fixed

#################################################################
# Motif look

MWMBorders
MWMButtons
MWMDecorHints
MWMFunctionHints
DecorateTransients

#################################################################
# (virtual) desktop

# Automatic desktop switching deactivated
EdgeScroll 0 0

# Delay in switching desktops
EdgeResistance 250 50

# Make windows being moved opaque
OpaqueMove 100

# Random positioning of new window
RandomPlacement

#################################################################
# Icon bar

ModulePath              /usr/X11/lib/X11/fvwm
PixmapPath              /home/prog/lib/xpm/
IconPath                /usr/include/X11/bitmaps/

Style "Fvwm Pager" Sticky
Style "GoodStuff"  Sticky
Style "Console"    Sticky
Style "xcalc"      Icon xcalc.xpm
Style "xterm"      Icon terminal.xpm
Style "rxvt"       Icon terminal.xpm
Style "*emacs"     Icon editor2.xpm
Style "*anual*"    Icon xman.xpm
Style "Mail"       Icon sndmail.xpm

#################################################################
# startup/restart initialization

Function "InitFunction"
         Exec    "I"     xsetroot -solid LightSlateGray
        Module  "I"     GoodStuff
        Module  "I"     FvwmPager 0 1
        Exec    "I"     xmmount -geometry -87+1 &
        Exec    "I"     xterm -sb -sl 300 -geometry +8+235 -iconic &
        Exec    "I"     xterm -sb -sl 300 -geometry +540+235 -iconic &
        Exec    "I"     xterm -sb -sl 300 -geometry +8+595 &
        Exec    "I"     xterm -sb -sl 300 -geometry +540+595 &
        Wait    "I"     xterm
EndFunction

Function "RestartFunction"
         Exec    "I"     xsetroot -solid LightSlateGray
        Module  "I"     GoodStuff
        Module  "I"     FvwmPager 0 1
EndFunction
```

```
#####################################################################
# Menu definitions

Popup "Shells"
        Title   "Shells"
        Exec    "MXterm"        exec mxterm &
        Exec    "Rxvt"          exec rxvt &
EndPopup


# Provides a list of modules to fire off
Popup "Modules"
         Title      "Modules"
         Module     "GoodStuff" GoodStuff
         Module     "FvwmIdentify"          FvwmIdent
         Module  "SaveDesktop"   FvwmSave
         Module  "Pager"         FvwmPager 0 1
         Module  "FvwmWinList"   FvwmWinList
EndPopup


# Programs in the root menu
Popup "Programs"
        Title   "Programs"
        Exec    "Xterm"         exec xterm &
        Popup   "Shells"        Shells
        Popup   "Modules"       Modules
        Exec    "Screen Saver"  exec xlock -nolock -mode flame &
        Nop     ""
        Restart "Restart Fvwm"  fvwm
        Quit    "Exit Fvwm"
EndPopup


# Menu with most common window operations
Popup "Window Ops"
         Title      "Window Ops"
         Move    "&Move          Alt+F7"
         Resize  "&Size          Alt+F8"
         Lower   "&Lower         Alt+F3"
         Raise   "Raise"
         Stick      "(Un)Stick"
         Iconify "(Un)Mi&nimize Alt+F9"
         Maximize "(Un)Ma&ximize           Alt+F10"
         Maximize "(Un)Maximize Vertical" 0 100
         Nop     ""
         Delete  "&Close         Alt+F4"
         Destroy "&Destroy"
EndPopup

#####################################################################
##
## Module definitions
##
#####################################################################

#################### GoodStuff icon bar ###################
# colors

*GoodStuffFore Black
*GoodStuffBack gray60

# Font
*GoodStuffFont -adobe-helvetica-medium-r-*-*-12-*-*-*-*-*-*-*

# Geometry
*GoodStuffGeometry -1+0

# One column
*GoodStuffColumns 1

# Two programs integrated in the icon bar
*GoodStuff junk xclock.xpm Swallow "xclock" xclock -bg gray60 -padding 3 &
*GoodStuff junk xload.xpm Swallow "xload" xload -bg gray60 &
```

```
# Several utilities
*GoodStuff XTerm    terminal.xpm Exec "xterm"      xterm &
*GoodStuff Xcalc    rcalc.xpm    Exec "Calculator" xcalc &
*GoodStuff Xman     xman.xpm     Exec "Manual"     xman -bothshow -notopbox &
*GoodStuff Mail     sndmail.xpm  Exec "pine"       xterm -T Mail -e pine &
*GoodStuff Emacs    editor2.xpm  Exec "emacs"      emacs &
```

9.9 Memory optimization

For computers with little memory and hard disk space available, such as notebooks, there are special economy versions of the X Window System for Linux. These usually sacrifice color display and resort to the XFree monochrome server. In addition, the number of X clients and fonts is reduced to the essential. The

Tiny X

TinyX package requires less than 4 MB of hard disk storage and, together with clients such as `fvwm` as window manager and `rxvt` as a substitute for the normal `xterm`, permit working with the graphical user interface with only 4 MB of RAM.

Chapter 10

Languages & Tools

One particular feature of Linux is the large number of freeware programming languages contained in binary format in the various Linux installation packages. These freeware tools seldom take a back seat to proprietary systems. This chapter introduces the various compilers, interpreters, and program development tools that are available under Linux and identifies their most important features.

10.1 Languages

C and UNIX

Undisputedly, C holds the position of the most important programming language for UNIX systems. Today's UNIX itself and all its utilities were developed largely in C. So it is not surprising that a C compiler serves as a central element of a UNIX development environment.

C compiler

As a rule, proprietary UNIX systems have included a C compiler in their packages. However, this has changed recently. Newer PC UNIX implementations are now being marketed in user versions without a C compiler and without network support. The full version including these components costs significantly more.

language spectrum

Programming language supported under Linux range from C, C++, and Objective C to Lisp and Prolog and on to Smalltalk and Forth. Even classical languages like FORTRAN and APL are available under Linux (whereby the FORTRAN product is only a FORTRAN-to-C converter that proves inadequate for professional requirements). Even BASIC programs can be developed

with either of two interpreters. A Modula-3 compiler is close to completion.

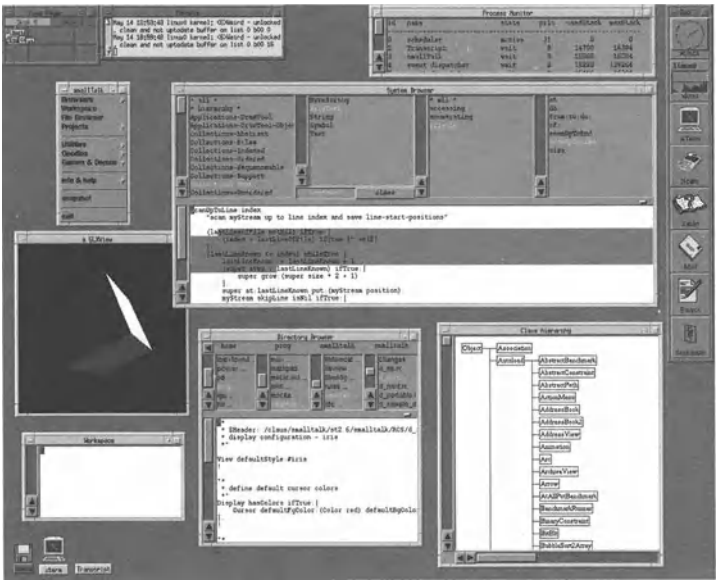


Fig. 10.1 Smalltalk/X for Linux

GNAT & Ada

GNAT, the Ada translator for Ada9X developed at New York University, is also available for Linux. It offers a complete compiler that uses the GCC back end. Since it became available before the official completion of the Ada9X standard, it will not be granted validation just yet.

In the near future, we can anticipate that almost every language will be available under Linux.

10.2 C/C++ compiler

gcc

The C compiler used under Linux is the GNU C Compiler (`gcc`) of the Free Software Foundation, which can generate optimized code. `gcc`'s availability for numerous UNIX platforms facilitates the porting of software. The GNU C Compiler supports ANSI and K&R standards for C as well as C++ and Objective C. This

compiler's detailed error messages on syntax errors deserve special note.

10.3 Pascal, Simula, and Modula-2

converters

The GNU C compiler also provides the basis for the Pascal-, Simula-, and FORTRAN-to-C converters. These converters read existing Pascal, Simula, or FORTRAN source code and generate a C program, which is then compiled with the C compiler. Skillful integration in the make system can leave the intermediate stage in C completely unnoticed by the programmer.

Beyond standard Pascal, the Pascal-to-C converter is even able to translate dialects of several Pascal variants, such as Turbo Pascal up to Version 5 and Macintosh Pascal.

Modula-2

The German Association for Mathematics and Data Processing (Gesellschaft für Mathematik und Datenverarbeitung, GMD) has ported their Modula-2 development environment MOCKA to Linux. It is interesting that the compiler itself was implemented in Modula-2; its source code is included in the package. In contrast to the above tools, MOCKA generates object code directly. MOCKA also supports linking C routines as *foreign modules*.

10.4 Lisp/Prolog

Common Lisp, CLOS

Since the programming languages used in the area of knowledge-base systems and artificial intelligence play an important role at universities, several implementations have become available. Here Lisp is of particular importance because it achieved relatively broad propagation as the programming language of the GNU Emacs. One Common Lisp implementation with an object-oriented extension (a subset of CLOS) named clisp is included in most Linux distributions. The ftp server `swi.psy.uva.nl` contains the extensive Prolog implementation of the University of Amsterdam, which is easy to compile under Linux.

10.5 Tcl

Tcl is an interpreter language in the form of a C library. For example, Tcl can be integrated in applications that require a script language, and it can be extended by additional commands with C routines. However, complete applications can likewise be developed in Tcl. A Tcl compiler for increased performance is currently in the development stage.

Tcl features

Tcl affords all the usual structures of conventional programming languages, such as loops, conditional statements, and procedures. Additional features, in particular associative arrays and list processing constructs, enable extremely elegant and short programs.

There are two basic approaches to developing an application in Tcl: Firstly, an application developed in C can be extended with a script language using the Tcl interpreter. Tcl is then linked to the application by defining additional Tcl commands. This approach lends itself primarily to already existing programs.

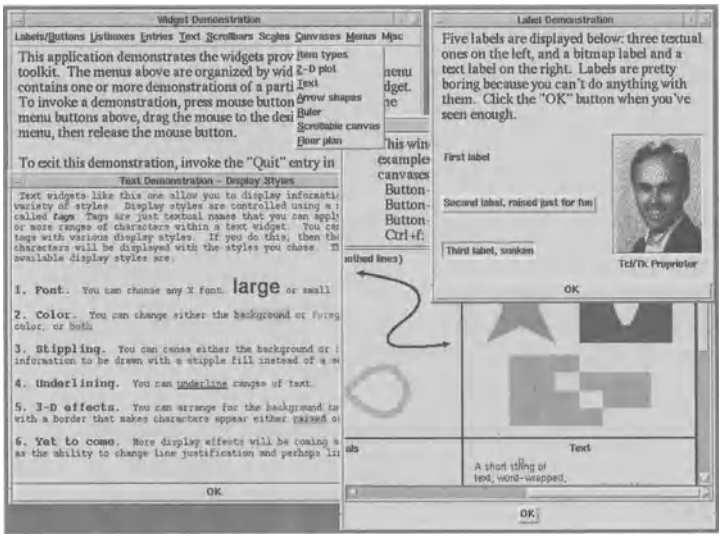


Fig. 10.2 Tcl/Tk demo program

Secondly, for a new design the main application could be implemented in Tcl. Functions that are not available can be

realized as Tcl extensions in C and invoked by Tcl. In this case the user interface could also be realized with the Tcl-based toolkit Tk.

Many Tcl language extensions are available in the public domain. These extensions enable access to SQL databases or UNIX system invocations. For example, Tcl-dp provides functions for network programming (sockets and RPC) and itcl extends Tcl by classes and methods for object-oriented programming. A debugger and a class browser are also available.

Tcl extensions

The ftp server `harbor.ecn.purdue.edu` at Purdue University in Indiana contains numerous such Tcl extensions as well as complete Tcl/Tk applications. World-Wide Web provides an overview of Tcl extensions, documents and programs. The address of this page is:

`http://web.cs.ualberta.ca/~wade/Auto/Tcl.htm`

10.6 awk, gawk

A traditional UNIX tool, `awk` permits the creation of smaller scripts for processing strings and text files. The name of the program stems from the initials of the three authors, Aho, Weinberger and Kernighan. Because the interpreter language of `awk` syntactically strongly resembles C, it is relatively easy to learn.

`awk` does not distinguish between numeric and string variables and it requires no variable declarations. For smaller projects `awk` can also serve as a prototyping tool. However, the size of an `awk` program should not exceed about 200 lines.

The following `awk` script adds the size of all files in the current directory and displays the total on the screen:

```
linux1:/> ls -l | awk '{sum += $5} END {print "Total : " sum}'
```

Rather than the original utility, Linux furnishes GNU `awk` (`gawk`), the implementation of the Free Software Foundation. (Commercial UNIX systems normally contain `awk` as well as an extended version called `nawk`.)

gawk

10.7 Perl

The interpreter language Perl combines the most important features of `sed`, `awk`, and the usual UNIX shells and proves especially suitable for processing text files. One advantage over classical tools is the possibility to open multiple files simultaneously.

Perl features

Perl permits the use of lists and associative arrays, with the size of the data structures limited only by available memory, which allows the processing of relatively large amounts of data. Naturally, Perl also has loops and other control structures, subroutines, recursion, and regular expressions. Various format statements for data output enable the creation of clear reports and tables. An integrated debugger allows the setting of breakpoints and stepwise execution of a program.

system administration

Perl proves suitable for system administration because it allows the production of scripts that run with root permission without creating flaws in the security system. It is also noteworthy that almost all routines of the standard C library and the UNIX kernels can be used directly under Perl, including functions for network programming via sockets.

An extension is currently being implemented that will allow the creation of graphical user interfaces and the use of object-oriented programming techniques.

10.8 Editors

GNU Emacs

A compiler alone does not constitute a full-scale development environment. At least an appropriate editor and a symbolic debugger are required. The GNU Emacs integrates these components. It can invoke numerous utilities like `grep` and RCS as well as the C compiler and the GNU debugger, so that the complete development cycle can run within the editor. Besides the GNU Emacs, the Linux system furnishes numerous other editors, which are introduced in Chapter 11.

Fig. 10.3 Software development with Emacs

10.9 GNU Debugger (GDB)

The GNU Debugger is a powerful symbolic debugger that supports C, C++, and Modula-2. It provides all functionality that developers expect from such a tool. Programs can be executed stepwise. Breakpoints allow the interruption of the execution at defined points, and watchpoints interrupt the program when specified values are modified. An option allows the continuous display of the values of selected variables or objects.

breakpoints, watchpoints

Newer versions of GDB also permit remote debugging, which means that the debugger can run on a different machine from the program to be debugged. The connection between the machines

remote debugging

can be either a serial interface or a network. Under Linux even the operating system can be analyzed with GDB, and running processes can be debugged.

graphical front ends

The GNU Debugger provides only a simple command-line-oriented user interface. However, there are graphical front ends that enable comfortable operation of the debugger with a mouse. These front ends start GDB as a second process and reroute its input and output. Thus they simulate user input and redirect debugger output to various windows. One example of such a graphical front end is `xxgdb`.

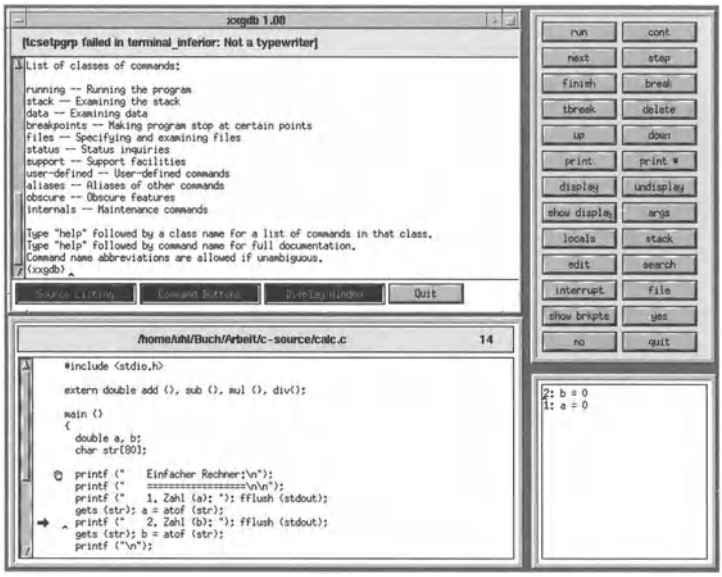


Fig. 10.4 Graphical front end for `gdb`

panes

The top pane displays the current position in the source code. The central area contains a number of command buttons that invoke basic debugger functions. The second pane from the bottom allows the user to enter textual commands when keyboard input seems more practical.

continuous display

GDB permits the continuous display of variable values in the bottom pane. An expression that occurs in the C source code can be evaluated easily: the user selects the expression in the source

code pane with the mouse; then the user clicks on the print button, and the result is displayed in the bottom pane.

The user can define a breakpoint similarly by positioning the text cursor at the appropriate position in the source code pane and pressing a button. Breakpoints are displayed with the symbol of a flat hand; a blue arrow marks the current position in the source code.

Another way to make the use of the GNU Debugger more comfortable is by integrating it with the Emacs editor. A special mode in Emacs permits debugging programs within a normal editor window, which is divided into two panes. The upper pane is for displaying the source code and marking the current position with an arrow. The lower pane is for controlling the debugger and for inputting commands. Using GDB within Emacs has some advantages for Emacs specialists, but it is certainly not as comfortable as using an X-based front end.

Emacs

10.10 Make utility

Another important component of the C development environment is the `make` utility. This command significantly simplifies the compilation of a project that consists of multiple modules. For this purpose, the programmer stores the dependencies between the individual program parts in a makefile. If the programmer modifies the source code of one of these modules, then only the modified parts and the modules dependent on them are recompiled.

makefile

The GNU variant of `make` used under Linux has several options that go beyond the normal `make` utility. For example, it supports the Revision Control System (RCS), a collection of commands for version management of source code. This system is described in more detail in the next section.

RCS

If a file cannot be found in the current directory, GNU `make` seeks a subdirectory named `RCS` from which the newest version of a module is automatically read. This even applies to the makefile.

10.11 Version control

The development of larger projects – consisting of a multitude of different modules and having new versions thereof evolving steadily – makes a version management system indispensable. Such a system gains in importance when multiple programmers are working on a project simultaneously.

SCCS

In addition to the Source Code Control System (SCCS) known from UNIX System V, Linux offers the more powerful Revision

RCS

Control System (RCS). Since all versions of a program in its development cycle are archived, if the need arises, a programmer can return to the source code of a long since outdated version. Normally the archive data are stored in a separate directory named `RCS` or `SCCS`.

RCS provides numerous commands, the most important of which we outline here.

- **ci** (check in) transfers the specified file into the `RCS` directory and thus freezes the current version. The complete version is not stored with each check-in, but only the difference with respect to the predecessor version, as determined with the `diff` command. The version number of the module is automatically incremented.
- **co** (check out) creates a nonmodifiable copy of the current version or of any predecessor of a module. If a module is to be modified, this must be indicated with the option `-l`. This protects the file from simultaneous modification by another programmer, for it can be checked out only once by only one user.
- **rlog** displays various information such as the status of an archive and the versions of the modules it contains.

Other commands permit the merging of two version branches and control certain options of the revision control system. To allow the version management and the rest of the development environment to work together smoothly, newer versions of the Emacs editor have a special mode for operating RCS.

10.12 XWPE

XWPE poses an interesting alternative to Emacs as a development environment. Users who are familiar with Borland products for DOS (Turbo C, Turbo Pascal, etc.), will soon feel at home, for XWPE greatly resembles them. Although XWPE is a text-oriented application, comfortable mouse operation is possible. XWPE's author also developed a terminal emulation program with color support for the X Window System. However, XWPE also runs on a normal terminal, although its look suffers considerably.

The integrated editor permits processing multiple texts in different windows. Both the compiler (gcc) and the debugger (gdb) can be started with either the keyboard or the mouse. Error messages are displayed in a separate window, and breakpoints can be defined directly in the editor.

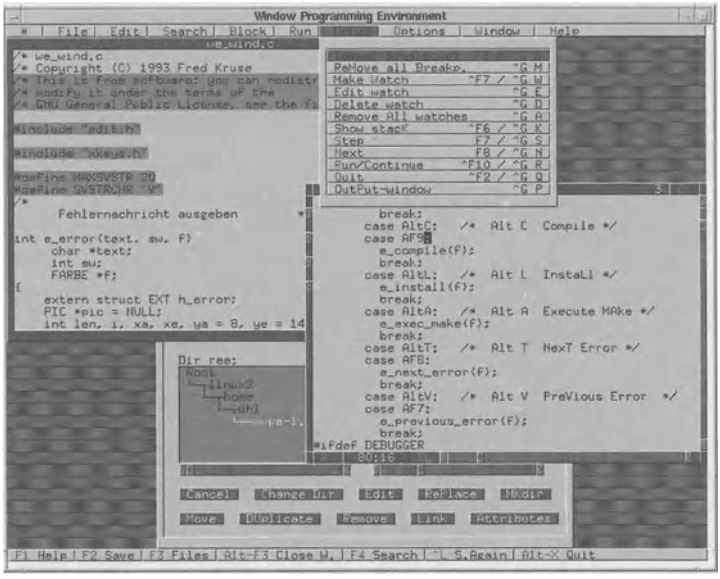


Fig. 10.5 XWPE as integrated development environment

XWPE can be procured from `ftp.rrzn.uni-hannover.de` in the directory `/pub/unix`.

10.13 Example

To give the reader a better impression of working with the above development environments, we introduce a small C program.

The program consists of multiple modules to add, subtract, multiply, and divide two numbers that are input. The main program contains only the two input prompts and the output of the results. The computations take place in separate modules. (Naturally, this simple example could have been combined in a single file.)

save

After the following source text has been input, it is stored as `main.c` with the key combination `<Ctrl-X> <Ctrl-S>`. This function could have been invoked from the menu instead.

```
#include <stdio.h>

extern double add (), sub (), mul (), div();

main ()
{
  double a, b;
  char str[80];

  printf ("    Simple Calculator:\n");
  printf ("    ==================\n\n");
  printf ("    1st number (a): "); fflush (stdout);
  gets (str); a = atof (str);
  printf ("    2nd number (b): "); fflush (stdout);
  gets (str); b = atof (str);
  printf ("\n");

  printf ("    a + b: %f\n", add (a, b));
  printf ("    a - b: %f\n", sub (a, b));
  printf ("    a * b: %f\n", mul (a, b));
  printf ("    a / b: %f\n", div (a, b));

  printf ("\n    Goodbye!\n");
}
```

Now the other modules can be entered. The key combination `<Ctrl-X> <Ctrl-F>` opens a new file. We can enter the file `addsub.c`.

```
double add (double a, double b)
{
  return a + b;
}

double sub (double a, double b)
{
  return a - b;
}
```

Then we enter the file `muldiv.c`.

```
Double mul (double a, double b)
{
  return a * a;
}

double div (double a, double b)
{
  return a / b
}
```

Now the project consists of three files. This also has an effect on the makefile:

makefile

```
CFLAGS = -g
OBJS = calc.o addsub.o muldiv.o

calc: $(OBJS)
       $(CC) -o $@ $(OBJS)
```

First the environment variable `CFLAGS` is set with the option `-g`, so that the compiler generates debug code. The newly defined variable `OBJS` contains the names of all object files after their declaration. This declaration only serves to increase the readability and is not absolutely necessary. The target is defined as the compiled and linked program `calc`. To generate the target, three object files are necessary. The following command links these object files after their compilation:

CFLAGS

OBJS

```
       $(CC) -o $< $(OBJS)
```

No separate rule needs to be specified for the translation of a `*.c` file to a `*.o` file . This is implicit in the make mechanism. On entry of the makefile, note that a `<TAB>` must precede the link command.

After the makefile has been saved, the complete project can be compiled and linked from the editor with the command `<Meta-X> compile`. The editor window splits. The lower window pane logs the commands that are executed and displays any error messages that arise. In our case we deliberately omitted a semicolon in the function `div` of the module `muldiv.c`. The compiler will detect this error when it compiles the source code. The command `<Meta-X> next-error` allows the processing of the error in the editor and places the cursor at the position in the

compile

source code where the error occurred. Entering the command <Meta-X> next-error again moves to the next error. After the errors have been corrected, <Meta-X> compile starts the compiler again. Naturally the programmer should assign each of the above commands to a keystroke combination to keep from having to enter the complete command each time. Such definitions are best placed in the file .emacs in the home directory.

debugger

If logical errors have found their way into the program, the GNU Debugger provides useful support in detecting them. Since the program was compiled with the compiler option -g, the debugger can be started immediately.

```
linux2:home/tul> xxgdb calc
```

breakpoints

First the required breakpoints need to be set. This can be done with the mouse or by entering the appropriate command. First the cursor is placed in the respective line in the text window where a breakpoint is to be set. A click on the break button of the Debugger then sets the breakpoint.

Sometimes it is easier to set a breakpoint via the command line, especially if the breakpoint is to be at the start of a command.

```
gdb> break main
```

The above command stops the program as soon as it reaches main.

After all breakpoints have been defined, the program can be started with the run command. When execution stops at a breakpoint, then it can be continued stepwise and the values of variables can be displayed.

```
gdb> print a
```

The above command displays the value of the variable a if it is within the current scope. The command display initiates the continuous display of variable values.

```
gdb> display b
```

The above command displays the value of variable `b` after each successive `gdb` command.

The debugged program can be started directly from Emacs. However, first a shell must be opened with the command `<Meta-X> shell`. Then all UNIX commands can be used within the editor. shell

The advantages of a shell that is executed within Emacs are the possibility to easily edit commands that have been executed and the ability to apply all editor commands, such as copying blocks, to the shell output.

10.14 Porting software

Only a small percentage of the programs that are available on the many ftp servers for UNIX systems have been especially ported to Linux. Since Linux adheres to the most important UNIX standards, however, compiling programs written for other UNIX systems generally poses no problem.

Unpacking

The programs on ftp servers have usually been stored as compressed and packed `tar` files. Such files were created with the `tar` program and can contain multiple files and directories. The name of the file usually contains information about its nature. If the name ends with `.tar.Z`, it is a `tar` archive that was compressed with the UNIX `compress` command. The extension `.tar.gz`, `.tgz` or `.taz` indicates that it is a `tar` archive that was compressed with the `gzip` program. tar files

To decompress and unpack such a file under Linux, it suffices to enter the following:

```
linux1:/home/tul> tar xvfz <file.tar.Z>
```

With the option z the tar program independently invokes the gzip program, which decompresses files that were compressed either with gzip itself or with the UNIX compress command.

shar files

Source codes that are sent as e-mails are often in shell archives whose names normally end with .shar. A normal Bourne shell suffices to extract the files from such an archive:

```
linux1:/home/tul> sh <File.shar>
```

Documentation

README

After unpacking an archive, the user should definitely read any included README or install files, which contain important instruction for compilation. These usually address the supported platforms and any problems that might arise.

Packages often contain extensive user documentation in the form of a PostScript or TeX file. At least a UNIX on-line reference manual page should be included in any case.

Makefile/Imakefile

options

The source code of a program is always accompanied by a makefile in which important options can be set, including compiler options, library paths, and the installation directory.

The makefile also aids the compilation and installation of the program. The invocations of make and make install start the compiler and the linker, respectively, and then installs the program. All these commands and options are explained in more detail in the README files that are included in almost all program packages.

imake file
xmkmf

The configuration of the X Window System software proves even easier. Instead of a makefile, X Window System programs normally contain an *i*makefile, which is a system-independent makefile. The command xmkmf automatically generates a makefile from the imakefile; the makefile contains all important paths and options of the respective system. For such programs it

usually suffices to enter the commands `xmkmf`, `make` and `make install`.

Configure

Newer FSF program packages often contain a script named `configure`. This script is invoked first, independently recognizes the operating system, and searches the system for its C compiler, other tools, and libraries. The information found is used to create one or more makefiles that contain the correct definitions. Sometimes the `configure` script even creates header files that are included in C program files.

automatic configuration

Manual adaptation

If the compiler terminates the make procedure with an error message, then the user must make some adaptations. This occurs frequently with programs that do not use the `configure` script. With a little practice, such adaptations are easy to carry out. The following is a potential error message:

```
linux2:/home/tul/tmp> gcc bsp.c
bsp.c: In function `main':
bsp.c:3: `errno' undeclared (first use this function)
bsp.c:3: (Each undeclared identifier is reported only once
bsp.c:3: for each function it appears in.)
linux2:/home/tul/tmp>
```

Here the compiler did not recognize the definition of a symbol. This can have several causes. If the symbol is a constant or a macro, then either it is not available under Linux or the header file in containing the declaration was not included with `#include`. The same applies to C functions whose prototypes are in header files. The user should first gain an overview of this function and its parameters from the on-line reference manual page.

To determine whether a function or another symbol exists under Linux, the system include files, i.e., the C header files of the Linux C library, can be searched. These header files are

header files

located in the directory `/usr/include` and can be searched with the following command:

```
find /usr/include -name "*.h" -exec grep "symbol" {} \; -print
```

find, grep

The `find` command seeks all header files in the directory `/usr/include` and all its subdirectories; then `grep` command searches each file for the symbol.

If the undefined symbol is a function that differs on many UNIX systems, then the source code frequently contains alternatives that can be activated with a compiler option such as `-DSYSV` or `-DBSD`. In the source code this could take the following form:

```
#ifdef SYSV
        function_a (x, y, z);
#else
        function_b (x, y, z);
#endif
```

#ifdef

If the user needs to make modifications in the source code, they should be embedded in `#ifdef linux` statements. Here `linux` is a symbol that is automatically declared by the compiler when it is invoked under Linux. The following is an example of conditional compilation for Linux variants:

```
#ifdef linux
        own modifications
#else
        old code
#endif
```

nm, grep

If the compiler does not report error messages, but the linker displays error messages on undefined symbols, then the program `nm` provides a valuable aid. It outputs a list of symbols declared and used in object files and libraries. In combination with using `grep` for searching the list of undefined symbols, this makes it possible to determine where a symbol is used and which library contains it:

```
linux1:/home/tul>nm prog1.o | grep "symbol"
```

With a little practice, the user will be able to adapt smaller programs in only a few minutes so that they can be compiled under Linux.

Archiving

After modifications have been made on the source code of a program, the modifications should be stored in a file with the `diff` (difference) command. Then the modifications can be applied at any time to the original program with the `patch` command, and these modifications also can be made available to other users.

diff

patch

10.15 Interface builder

Creating graphical user interfaces without appropriate tools can be outright troublesome. ParcPlace markets a very interesting program package, the Object Builder, for the interactive design of such interfaces.

Object Builder

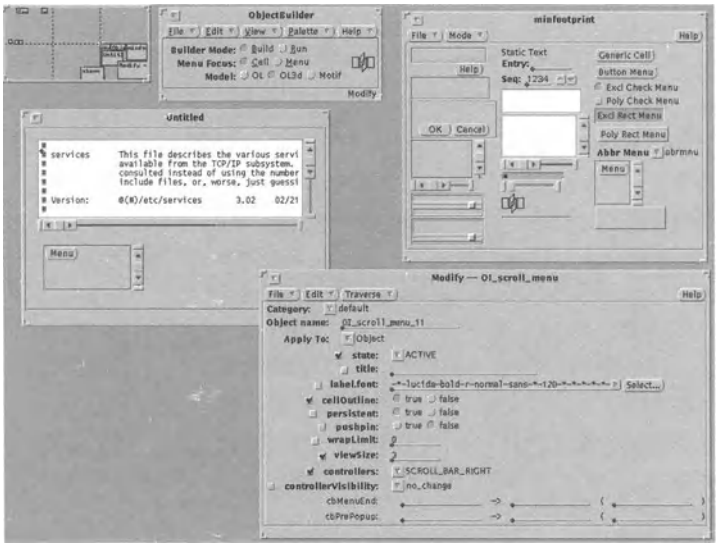


Fig. 10.6 ParcPlace's interface builder

This interface builder supports the manipulation of graphical interface objects with the mouse and generates C++ source code.

Object Library

In combination with the Object Library, graphical user interfaces can be generated for the OpenLook or the Motif convention. A command line option sets the preferred look and feel.

Both products are available for all common UNIX platforms. The purchase price runs into four figures. Only the Linux version is freeware. For serious applications, however, a programmer should have the handbooks, which are available from ParcPlace.

Applications

Because such a multitude of application programs have become available for Linux, we can only present a small selection here.

11.1 Working environment G.R.E.A.T.

The GRaphical Environment And deskTop (G.R.E.A.T.), developed by the Free Software Association Germany (FSAG) provides the user with a comfortable environment for managing all system components.

Fig. 11.1 G.R.E.A.T. environment

Important applications can be placed in an icon bar at the edge of the screen. Animated icons and sound provide attractive feedback. Files can be dragged from the file manager and dropped on the icon bar, which automatically starts the respective application. G.R.E.A.T. uses the OSF/Motif widget set and is intended to serve as an integration platform for future Linux applications.

11.2 Editors

Text editing programs are among the most important components of a computer system. Nearly a dozen such editors are available under Linux. Several of them are available on every UNIX system, while others have to be purchased separately with proprietary systems.

vi

command and input modes

The standard editor of every UNIX system is certainly `vi`. Since this is a rather aged program, it is no wonder that its user comfort leaves room for improvement in several points. `vi`'s distinction between command and input mode irritates most users. Without a doubt this approach has its advantages, but it also requires some adjustment. For example, it is possible to repeat the last command string and to replace the next three words in a step with a new word.

Since the source code of the `vi` editor is not available, a `vi` clone tends to be used under Linux. Even here there are two alternatives, `elvis` and `vim`, both of which emulate (almost) all commands of the original and provide several extensions as well.

sed

stream editor

Not an editor in the usual sense, `sed` is rather used as a stream editor to modify a data stream by means of commands in a control file. `sed`'s commands are largely compatible with those of

vi, but the user does not input them interactively; instead, they are read from a file or from the command line. sed often comes to play in UNIX shell scripts or for processing extremely large files that normal editors cannot load.

joe

joe is short for "Joe's own Editor". This editor finds appreciation among converts from DOS because its extended keys rely strongly on those of the well-known PC word processing program Wordstar and the Turbo Pascal editor. The command and editing modes are not separate. Marked text blocks are displayed inversely, which is not always a given under UNIX. joe can divide the screen into multiple windows, format paragraphs, and use hyphenation mode.

Wordstar keys

xedit

xedit is part of the X Window System package. Contrary to the above editors, xedit is not confined to an ASCII environment. Yet, although xedit runs under a graphical user interface, it does not afford the level of comfort that would be expected; its available commands are essentially limited to the possibilities given by the Athena text widgets. While this editor will not likely ever enjoy broad propagation, it does suffice for simple tasks.

axe

axe is another editor that runs only under the X Window System. It also uses the Athena text widgets, but provides much more functionality than xedit. The user can open an arbitrary number of text files in different windows, open and save files interactively via a file selection dialog, seek and replace text, and format paragraphs. Simple on-line help gives quick information on available commands. axe proves to be a multifaceted and user-friendly editor under Linux for the X Window System.

athena widgets

xcoral

`xcoral` also requires the X Window System environment. Although it uses no standard toolkit, `xcoral` has a pleasing look with a menu bar and a scroll bar. `xcoral` particularly interests C and C++ programmers because of its integrated function and class browsers. When `xcoral` is started, it scans all C/C++ files and displays all the identifiers contained therein in an alphabetical list. From the browser the user can then branch to the respective location in the source code. This facilitates in particular a programmer's familiarization with existing source code. For writing new programs, the programmer can select an optional mode that handles uniform formatting and can create function and class templates. Keyboard commands in `xcoral` bear a strong resemblance to the Emacs editor.

function/class browsers

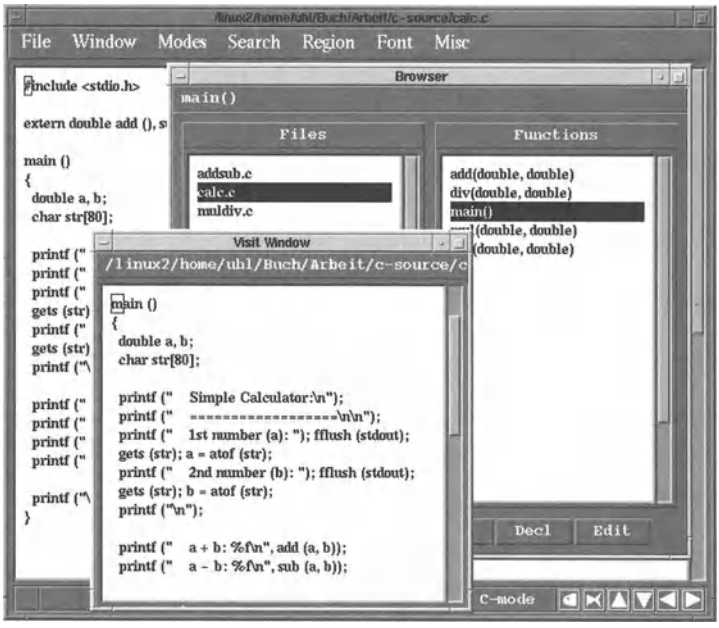


Fig. 11.2 Editor `xcoral`

asedit

`asedit` uses the Motif widget set, which is reflected particularly in its consistent interface. Although this editor offers the most important commands, such as loading and saving texts and search-and-replace, it is quite Spartan otherwise. Its hypertext help deserves special mention.

Emacs and variants

The most popular editor for UNIX is certainly Emacs in all its variants. A major reason for its popularity is that it runs both on pure ASCII terminals and in a graphical X11 environment. `Emacs` was developed by the founder of the Free Software Foundation, Richard Stallman, and is available for almost all UNIX platforms. Furthermore, special versions are available for VMS and DOS.

Emacs is a nonmodal, comfortable full-screen editor. The user can retain multiple texts in memory simultaneously and view these in separate or split windows. In regular intervals Emacs automatically creates a backup copy of all buffers and reorganizes memory.

Emacs was implemented largely in its own LISP, which makes the editor extraordinarily configurable. Assuming familiarity with LISP, the user can implement additional features. Numerous users have taken advantage of this possibility, so that an unbelievable number of Emacs extensions exist.

extensibility

Because it is so powerful, the Emacs package is also relatively extensive. The base installation requires about 20 MB of storage on the hard disk, but this includes a detailed hypertext help function that contains not just the description of the editor itself but of most GNU packages as well. Emacs' unlimited undo function deserves particular mention: every change of the buffer can be undone with a keystroke.

space requirements

hypertext help

undo

Emacs provides numerous major modes that adapt the editor optimally to the task at hand. Such modes support the programmer or author in formatting source code (or other text). For example, the text mode permits automatic word wrap.

major modes

In the C or Lisp mode, nested statements are automatically inset according to their nesting depth. When a closing bracket is entered, the cursor blinks briefly at the opening bracket to make the hierarchy clear. Keywords or comments can be displayed in another font. Emacs supports nearly every significant programming language.

dired mode

The dired mode permits the management of files and directories of the file system. Files can be deleted, copied, renamed, and loaded. A hexadecimal mode enables processing binary files. A mathematics program written in Emacs Lisp solves linear equation systems, reduces terms, and handles symbolic differentiation.

rmail and news modes

Beyond all this, Emacs can serve as a comfortable front end for existing applications. The special `rmail` mode permits the sending and receiving of mails; the `gnus` mode supports the management of news from USENET.

Emacs' interface to the GNU C compiler and debugger are of particular interest to programmers. Emacs can thus serve as a development shell. On a keystroke, the current text can be compiled. The compilers' output is redirected to a separate window. Furthermore, the programmer can jump from one syntax error to the next in the source code and start the debugger if needed.

Fig. 11.3 Emacs version 19

Emacs is extremely configurable. Each key can be bound to any LISP function. If Emacs is running in a graphical environment, then it provides an extensible menu bar as well as a scroll bar that indicates the current position in the text (as of version 19). Furthermore, individual text elements can be highlighted with a special color or a different font. configuration

But Emacs also provides the user with some entertainment. In addition to the animated solution to the Towers of Hanoi, a fun-loving user can enjoy the question and answer session of Weizenbaum's electronic psychiatrist ELIZA. eliza

With the release of Emacs version 19, other Emacs versions such as `xemacs` and `epoch` declined in importance. Hence we refrain from giving further details on these.

11.3 Graphic programs

Numerous painting and drawing programs enable the creation of graphics and the processing of pictures. Most of these tools support a broad range of graphic formats, so that data exchange between DOS and Linux systems really should be no problem.

xv

J. Bradley's xv, a very powerful program for the display and manipulation of graphics of all kinds, enjoys wide dissemination in the UNIX world. It has a very matured interface and is easy to use despite its many powerful features.

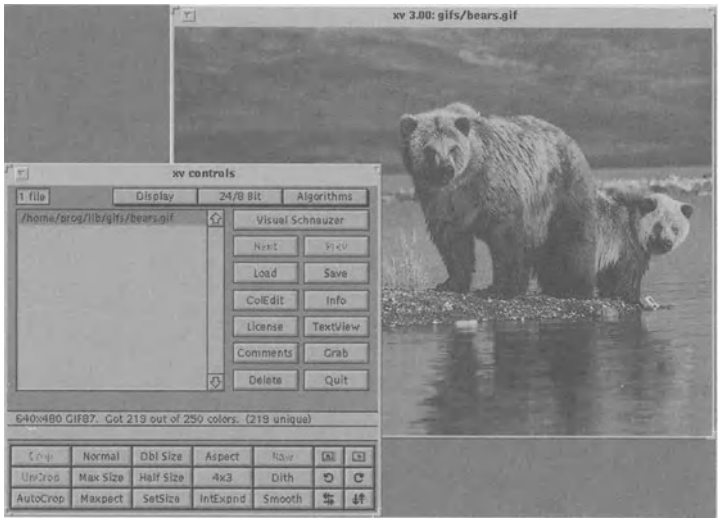


Fig. 11.4 xv with file browser

xv loads and saves all common graphic formats such as GIF, TIFF, PostScript, JPEG, and PBM. A graphic to be loaded is selected from a file selection menu or (since version 3.00) via an integrated file manager that can display selected files as mini-pictures.

size, brightness, contrast and tone

The size of a loaded graphic can be changed freely. A graphic processing menu permits the modification of numerous parameters such as brightness, contrast, and tone by means of

numerous buttons and dial controls as well as mouse-modifiable curves.

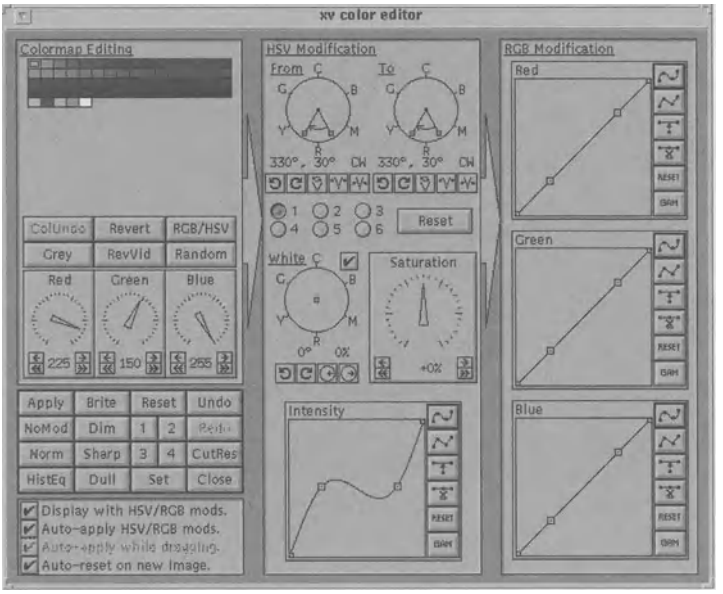


Fig. 11.5 xv color editor

A graphic loaded into xv can be modified in various ways, such as by using an algorithm like "oil painting" or "emboss". Furthermore, it can be displayed in various forms as a background graphic which is retained after quitting the program.

emboss

xfig

A program for creating vector graphics under Linux, xfig offers all the usual drawing tools as well as embedding text in various fonts. One of its interesting features is the export of graphics in many formats. Besides its own fig format, xfig supports standards like PostScript, HPGL, and TeX in variants.

many formats

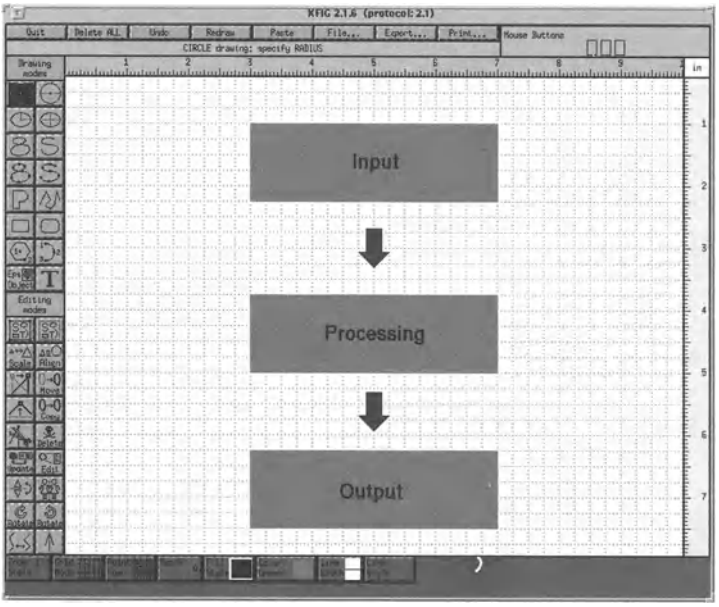


Fig. 11.6 Drawing program `xfig`

idraw

InterViews

Stanford University's InterViews package contains a quite powerful vector-oriented drawing program named `idraw`. Besides the usual graphical elements such as lines, rectangles, circles, and Bezier curves, the free rotation of texts in various fonts is supported.

The only file format that this program supports is PostScript. Unfortunately the program places high demands on the hardware, which makes the program uninteresting for 386 computers.

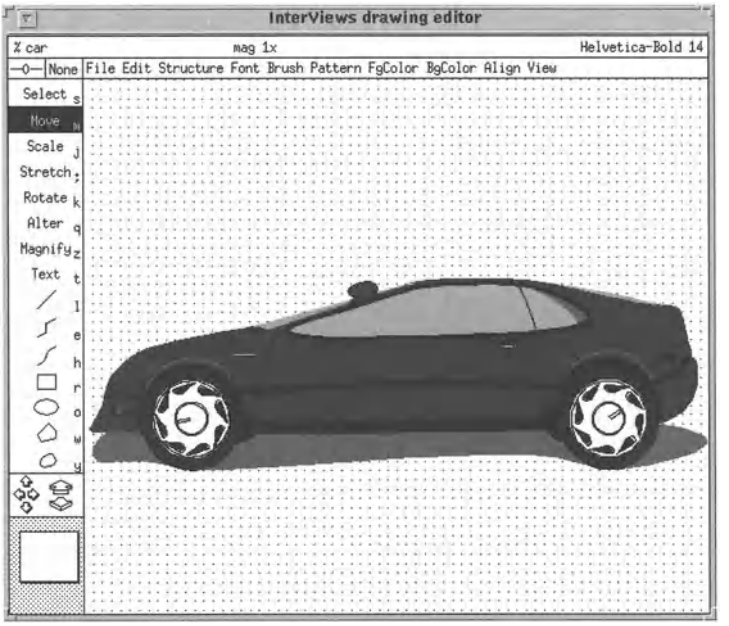


Fig. 11.7 Drawing program `idraw`

xpaint

`xpaint`, a program for the creation and manipulation of graphics, borrows from the MacDraw program on the Apple Macintosh. Multiple graphics can be processed in different windows. Various tools support the drawing of any kind of figures, with simple elements such as lines and circles as well as freehand curves being supported.

A fill-pattern editor permits the creation of additional patterns. The adjustable zoom function proves quite practical; it displays an excerpt from the graphic in a separate window. It is noteworthy that all drawing functions are supported in the zoom window as well. The `xpaint` font browser offers a clear selection of fonts, including scaleable fonts as available under X11 R5 and R6.

zoom

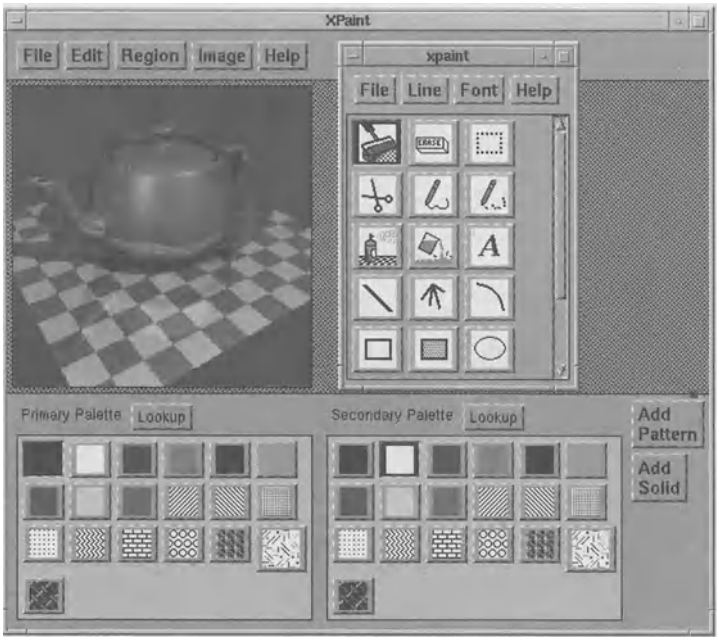


Fig. 11.8 Painting program `xpaint`

Ghostscript/Ghostview

PostScript interpreter

An important component of the Linux system is the PostScript interpreter Ghostscript (`gs`). Together with its graphical front end Ghostview it permits the comfortable display of PostScript files. But Ghostscript implements more than just the screen output; it also transforms an ordinary dot-matrix or ink-jet printer into a full PostScript printer. When color graphics are printed on a monochrome device, Ghostscript uses dithering to achieve reasonable quality.

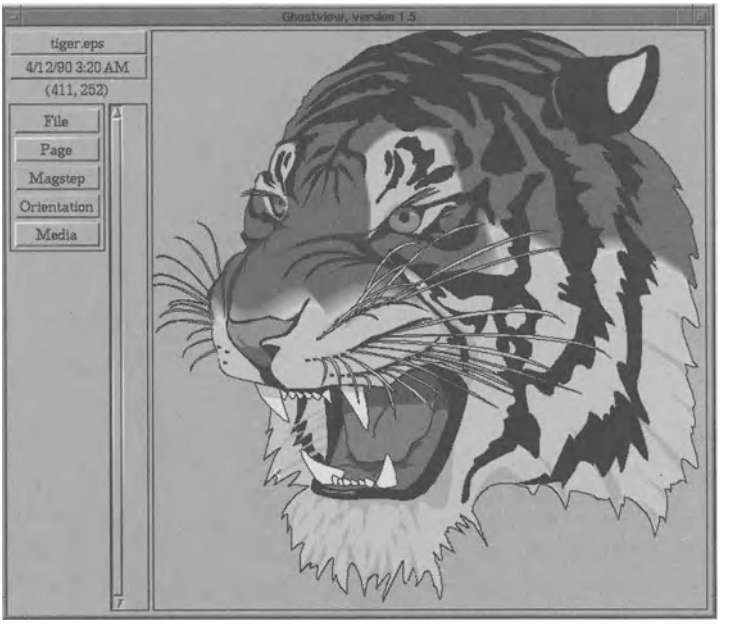


Fig. 11.9 PostScript interpreter `ghostview`

Currently Ghostscript can only output files that are PostScript level 1 compatible, but level 2 support has been announced. To be able to handle text that uses any of the copyrighted Adobe fonts, Ghostscript includes a package of similar fonts, but their quality is noticeably poor. Fortunately Ghostscript also allows the installation and use of the original fonts.

PostScript

MPEG Video Player

Any of the many MPEG videos from the Internet can be viewed under Linux using the MPEG Player from the University of California at Berkeley. Note that this requires no special hardware and that the output can be redirected to other screens. Unfortunately the performance of the computer and its graphic board strongly influences the speed of the player.

Berkeley

Fig. 11.10 MPEG video player

11.4 Word processing

Most text processing systems under Linux are not word processors. Generally control sequences are inserted appropriately into ASCII text, in order to influence the formatting of the text. This approach might seem quite inconvenient at first, but it does have its advantages, particularly for large documents.

groff

ASCII text

The classical way to create formatted documents under a UNIX system is to process an ASCII text for the command `nroff`. This is a package for formatting texts, tables, formulas, and simple graphics. The user enters text in ASCII format and influences the layout with corresponding format commands in the text. Output can be on a text-oriented or a graphical device.

All UNIX on-line reference manual pages were created in this way. Normally the primary use of `nroff` is to display manual pages. `nroff`'s functionality can be extended with external macros, and macro packages are available for various tasks. There is a dedicated macro file (`an`) for formatting a manual page. The command looks like this:

```
linux2:/home> nroff -man ls.1 | more
```

Linux provides `groff`, an enhancement, rather than the original `nroff`. The command `nroff` is only a script that invokes `groff` with the correct parameters for ASCII output. If `groff` is invoked directly without these parameters, then PostScript text is output. This proves practical for printing reference manual pages on a PostScript-compatible printer, since the text is properly formatted and output with various fonts. The following terminal capture shows the creation of a formatted PostScript file from an on-line reference manual page.

```
linux1:/root# groff -man /usr/local/man/scotty.1 >scotty.ps
```

TeX

The typesetting system TeX (pronounced teck) by Donald E. Knuth represents a world of its own. On first contact, the user of a modern WYSIWYG word processing system will feel projected into the stone age of computers. Nevertheless, TeX affords some features that make it superior to normal word processing in several ways. One advantage of the system is its free availability for almost all computer platforms and thus the portability of the generated texts.

TeX processes files in ASCII format that contains special formatting commands. The powerful program proves especially suited for typesetting mathematical material, but the quality of normal texts also exceeds that of word processing programs. TeX translates an ASCII input file to a DVI (device-independent) file, which can be displayed on the screen or printed. The format of the DVI file is identical on all computer systems, which makes transferring a TeX file easy. The file can also be output to a linotronic machine.

DVI format

Numerous macro packages and auxiliary programs exist for TeX that are also available under Linux. These include a graphical previewer (`xdvi`), a utility for sorting an index file, and a program to automatically generate missing fonts. There are also many drivers that convert DVI files to PostScript (`dvips`) or output them on non-PostScript printers (dot-matrix, ink jet, laser).

Previewer

Graphics can be embedded with LaTeX commands or with externally created PostScript files.

LaTeX

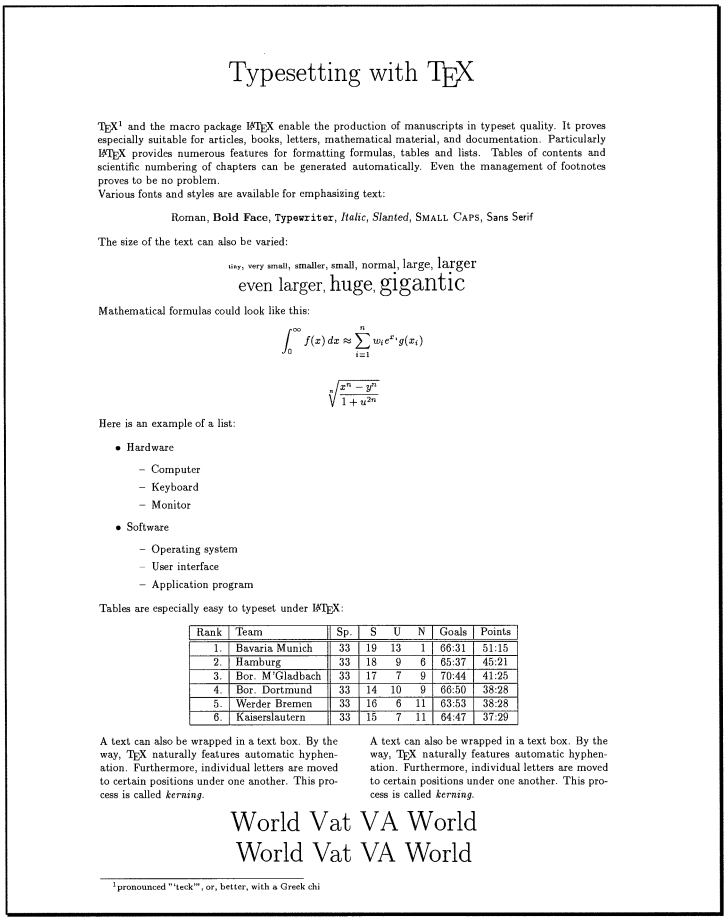
One of the best-known macro packages is LaTeX, by Leslie Lamport, which appreciably eases working with TeX. Naturally there are also version for non-English texts. LaTeX supports the automatic generation of a table of contents and an index and eases the formatting of tables and lists.

The following example shows a LaTeX input file followed by its output:

```
\documentstyle{article}
\topmargin -15mm \headsep 0mm \textwidth 16cm
\textheight 26cm \oddsidemargin 0cm \parindent 0mm

\begin{document}
\thispagestyle{empty}

\centerline{{\Huge Typesetting with \TeX}}
\vspace{1cm}

\TeX\footnote{pronounced "`teck'", or, better, with a Greek chi} and
the macro package \LaTeX\/ enable the production of manuscripts in
typeset quality. It  proves especially suitable for articles, books,
letters, mathematical material,  and documentation. Particularly
\LaTeX\/ provides numerous features for  formatting formulas, tables
and lists. Tables of contents and scientific  numbering of chapters can
be generated automatically. Even the management of  footnotes proves to
be no problem. Various fonts and styles are available for emphasizing
text:

\begin{center}
{\rm Roman}, {\bf Bold Face}, {\tt Typewriter}, {\it Italic},
{\sl Slanted}, {\sc Small Caps}, {\sf Sans Serif}
\end{center}

The size of the text can also be varied:

\begin{center}
{\tiny tiny}, {\scriptsize very small}, {\footnotesize smaller},
{\small small}, {\normalsize normal}, {\large large}, {\Large larger}\\
{\LARGE even larger}, {\huge huge}, {\Huge gigantic}
\end{center}

Mathematical formulas could look like this:
\begin{displaymath}
        \int_0^\infty f(x)\,dx \approx \sum_{i=1}^n w_i e^{x_i} g(x_i)
\end{displaymath}

\begin{displaymath}
        \sqrt[n]{\frac{x^n - y^n}{1 + u^{2n}}}
\end{displaymath}

Here is an example of a list:

\begin{itemize}
\item Hardware
\begin{itemize}
\item Computer \item Keyboard \item Monitor
\end{itemize}
\item Software
\begin{itemize}
\item Operating system \item User interface \item Application program
\end{itemize} \end{itemize}
```

```
Tables are especially easy to typeset under \LaTeX\/:

\begin{center} \begin{tabular}{|r|l||c|rrr|c|c|} \hline
Rank & Team            & Sp. & S  & U  & N  & Goals & Points\\
\hline\hline
1.  & Bavaria Munich  & 33  & 19 & 13 & 1  & 66:31 & 51:15\\ \hline
2.  & Hamburg         & 33  & 18 & 9  & 6  & 65:37 & 45:21\\ \hline
3.  & Bor. M'Gladbach & 33  & 17 & 7  & 9  & 70:44 & 41:25\\ \hline
4.  & Bor. Dortmund   & 33  & 14 & 10 & 9  & 66:50 & 38:28\\ \hline
5.  & Werder Bremen   & 33  & 16 & 6  & 11 & 63:53 & 38:28\\ \hline
6.  & Kaiserslautern  & 33  & 15 & 7  & 11 & 64:47 & 37:29\\ \hline
\end{tabular} \end{center}

\newcommand{\absatz}{
\begin{minipage}[b]{7.5cm}
A text can also be wrapped in a text box. By the way, \TeX\/ naturally
features automatic hyphenation. Furthermore, individual letters are
moved to certain  positions under one another. This process is called
{\em kerning}.
\end{minipage}}

\absatz \hfill \absatz

\begin{center}
{\Huge W\/orld V\/at V\/A W\/orld}
\end{center}
\begin{center}
{\Huge World Vat VA World}
\end{center}

\end{document}
```

After compilation the result can be displayed with the command `xdvi` with the following results:



Typesetting with TEX

TEX[1] and the macro package LATEX enable the production of manuscripts in typeset quality. It proves especially suitable for articles, books, letters, mathematical material, and documentation. Particularly LATEX provides numerous features for formatting formulas, tables and lists. Tables of contents and scientific numbering of chapters can be generated automatically. Even the management of footnotes proves to be no problem.
Various fonts and styles are available for emphasizing text:

Roman, **Bold Face**, Typewriter, *Italic*, *Slanted*, SMALL CAPS, Sans Serif

The size of the text can also be varied:

tiny, very small, smaller, small, normal, large, larger
even larger, huge, gigantic

Mathematical formulas could look like this:

$$\int_0^\infty f(x)\,dx \approx \sum_{i=1}^{n} w_i e^{x_i} g(x_i)$$

$$\sqrt{\frac{x^n - y^n}{1 + u^{2n}}}$$

Here is an example of a list:

- Hardware
  - Computer
  - Keyboard
  - Monitor
- Software
  - Operating system
  - User interface
  - Application program

Tables are especially easy to typeset under LATEX:

Rank	Team	Sp.	S	U	N	Goals	Points
1.	Bavaria Munich	33	19	13	1	66:31	51:15
2.	Hamburg	33	18	9	6	65:37	45:21
3.	Bor. M'Gladbach	33	17	7	9	70:44	41:25
4.	Bor. Dortmund	33	14	10	9	66:50	38:28
5.	Werder Bremen	33	16	6	11	63:53	38:28
6.	Kaiserslautern	33	15	7	11	64:47	37:29

A text can also be wrapped in a text box. By the way, TEX naturally features automatic hyphenation. Furthermore, individual letters are moved to certain positions under one another. This process is called *kerning*.

A text can also be wrapped in a text box. By the way, TEX naturally features automatic hyphenation. Furthermore, individual letters are moved to certain positions under one another. This process is called *kerning*.

World Vat VA World
World Vat VA World

[1] pronounced "'teck'", or, better, with a Greek chi

Fig. 11.11 TeX output

Andrew User Interface System (AUIS)

Texts can be formatted much more easily than in LaTeX with the `ez` editor of the Andrew User Interface Systems. Text passages can be enhanced with bold, various fonts, and underlining. `ez` permits direct WYSIWYG representation of the documents and the embedding of graphics, spreadsheets, formulas and animations. Dedicated editors are furnished for each of these types of embedded objects. A special multimedia mail supports the transfer of such files. Since the AUIS environment is rather

extensive, it requires about 30 MB of hard disk space. In contrast, the `ez` editor can make do with only 5 MB.

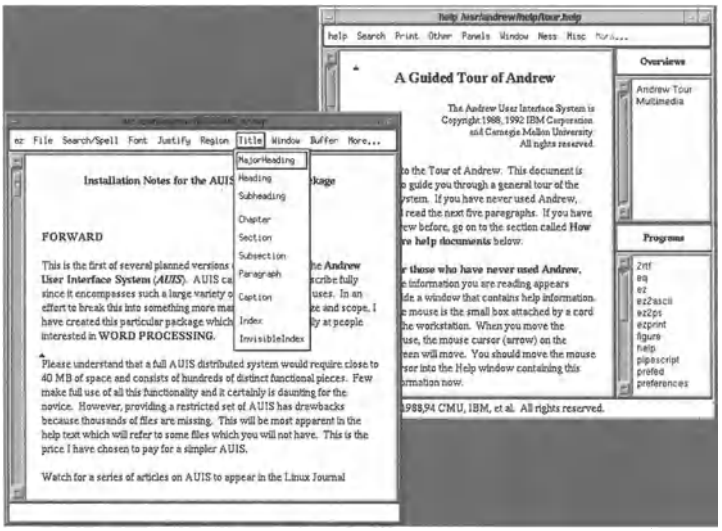


Fig. 11.12 Word processor `ez`

11.5 Games

To relax after a day's work, the user can enjoy one of the many games available under Linux. For example, the extremely popular game Tetris is available under Linux in a very attractive form.

GNU Chess

For more challenging entertainment, have a look at `xboard` and GNU Chess. The chess skill of this program is astonishing. At chess competitions GNU Chess has defeated commercial chess programs. GNU Chess itself is not particularly user-friendly, so it requires the graphical front end `xboard` to make playing practical. `xboard` can also start GNU Chess twice and let the games play against each other. Another option allows chess

tournament tested

parties on the Internet, making it possible to find an adequate partner somewhere in the world at any time, day or night.

MUDs and Crossfire

Games that can be played on a network by multiple participants prove especially interesting. One widespread form of such network games is MUDs (Multi-User Dungeons). Each player logs in to a common server via telnet and can use commands like `say`, `get`, or `go`, and can move. MUDs are normally limited to textual input and output.

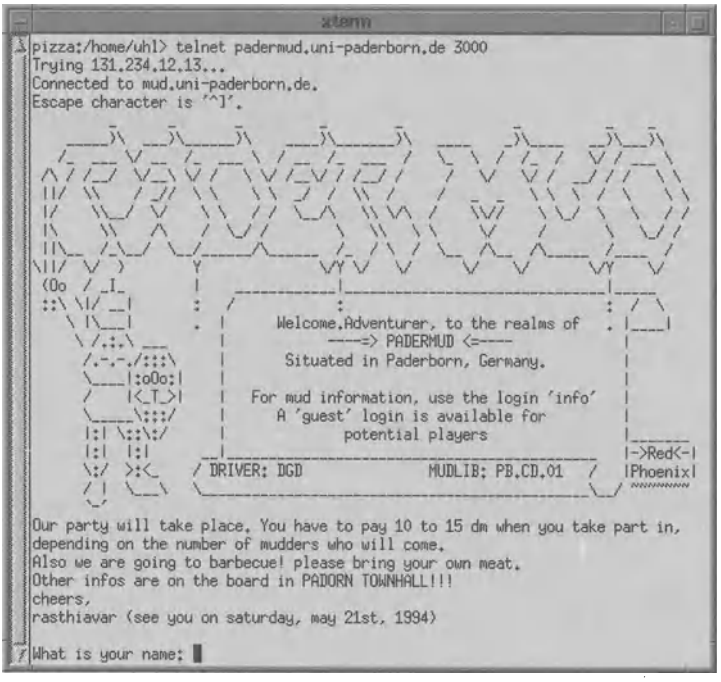


Fig. 11.13 Multi-User Dungeon Game (MUD)

Some addresses of MUDs that can be reached by telnet are :

- morgen.cs.tu-berlin.de Port 7680
- padermud.uni-paderborn.de Port 3000
- pascal.uni-muenster.de Port 4711

Further information is available in the many newsgroups on MUDs and in the MUD FAQ (see page 138).

A leap ahead of the text-oriented MUDs, the multi-user role play Crossfire requires special client software that graphically depicts the virtual world in which the player moves. The source code of the game can be copied from the ftp server `ftp.ifi.uio.no` in the directory `/pub/crossfire`.

virtual world

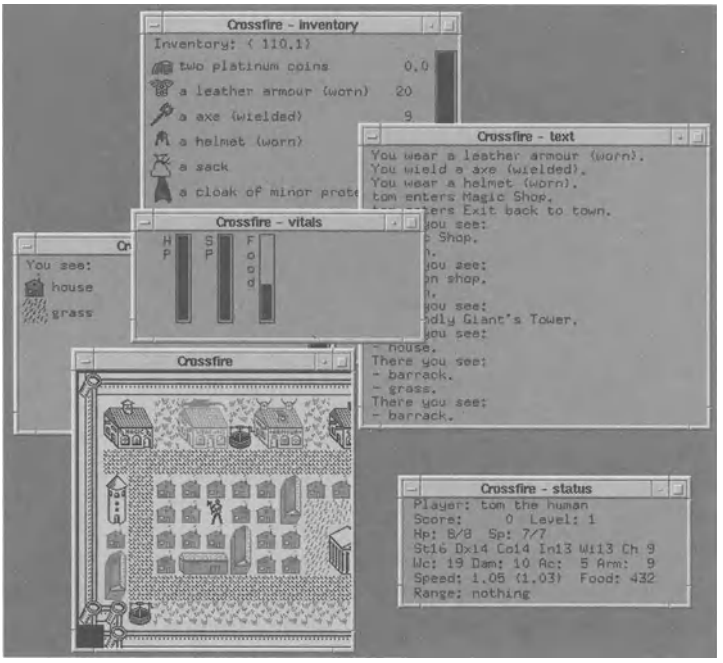


Fig. 11.14 Graphical adventure game `Crossfire`

11.6 Other applications

This section serves as a catch-all for some practical utilities and programs that do not fit into the above categories.

XVMount

CD-ROM

user mount

A practical utility that simplifies everyday transactions with diskettes, removable hard disks, and CD-ROMs is `xvmount`. To be able to access a new file system, the user must first mount it in the existing file tree with the command `mount`. `xvmount` permits the normal user to mount certain file systems. Normally, to mount a file system, its type must also be specified. `xvmount` detects the type automatically so that the user need not be bothered. A user-friendly graphical user interface contributes to ease of use.

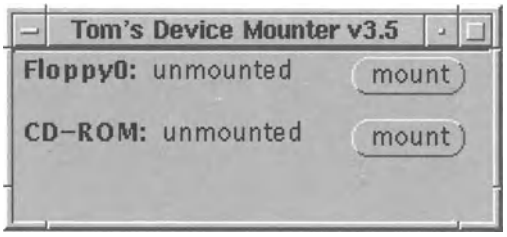


Fig. 11.15 `xvmount` utility

Seyon

terminal emulation

To transfer data or to log in to another system via a modem, the user needs a terminal program. Seyon is such a program that uses `xterm` as its terminal emulator, yielding a full-scale VT100 terminal emulation. Since the program has an X11 user interface, the interface parameters can be set comfortably with the mouse. The program provides both a phone list and a simple command interpreter. When receiving files with the zmodem protocol, the receiving machine starts the corresponding receiving program

(rz) automatically. Optionally, all screen output can be logged in a file.

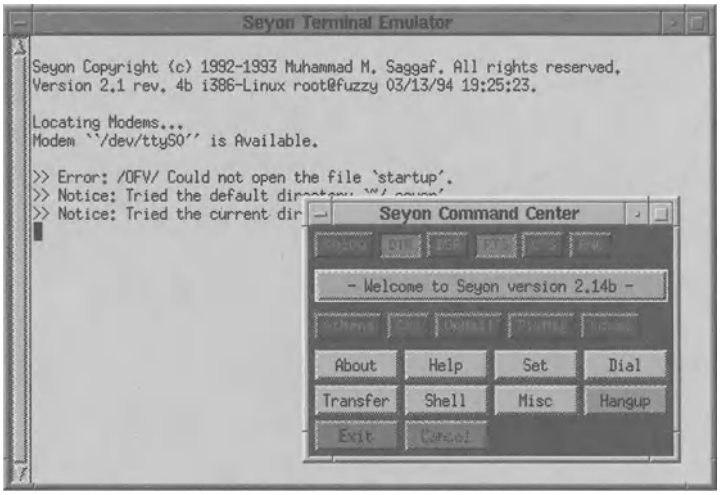


Fig. 11.16 Terminal program seyon

X file manager

Many UNIX users find the command line environment of the usual shells for file management inadequate. Despite the broad range of features of UNIX shells, they prefer graphical user interfaces. One relatively new program for the graphical representation of directory trees and for copying, renaming, and moving files is the X file manager. Using multiple windows, the user can gain an overview of the file structure and can copy files and directories using the mouse. Programs that are used especially often can be placed in a separate window. Clicking on a file starts the associated program for processing or displaying it. Files and directories are depicted as icons that can be in color. The user can individually determine the correspondence of file types to icons.

directory trees

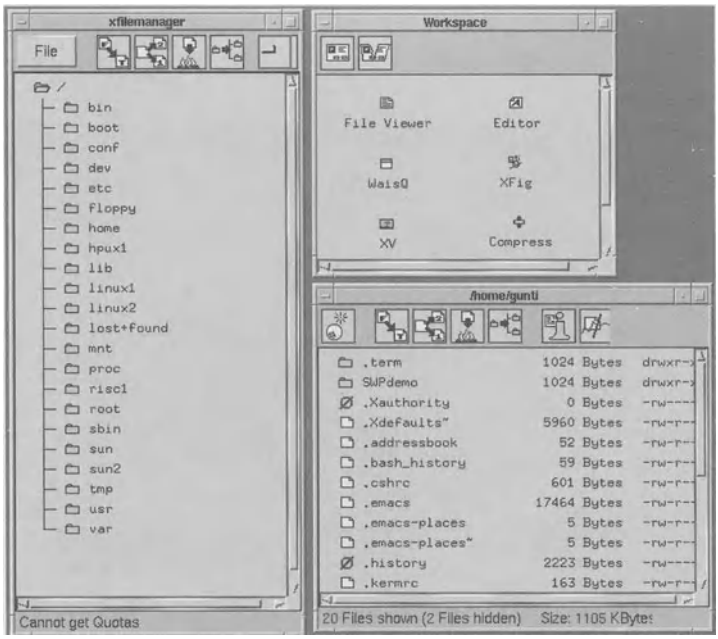


Fig. 11.17 File manager `xfilemgr`

Ingres and Postgres

limited versions

The database system Ingres and its successor Postgres from the University of California at Berkeley have been ported to Linux. However, do not confuse this version of Ingres with the commercial SQL database system. The Linux version is significantly older and has a simpler query language instead of SQL. Postgres is an object-oriented database, but, unlike many of the proprietary systems, it is not based on C++ and persistent objects, but has its own peculiar approach. Both systems are quite interesting for educational purposes, but their usefulness is limited for developing commercial applications.

MuPAD

A particularly interesting program package for students is MuPAD. This computer algebra system was developed at the University of Paderborn (Germany) and is furnished free of charge to noncommercial institutions. MuPAD supports the user in solving various types of mathematical problems. An integrated programming language enables the implementation of the user's own algorithms.

computer algebra

MuPAD was especially designed to handle parallel problems. Therefore future versions will be able to take efficient advantage of multiprocessor architectures and workstation clusters.

MuPAD provides its own debugger with a graphical front end. MuPAD itself has a simple command-line-oriented user interface. However, an OpenLook-based front end makes the program appreciably easier to use. MuPAD's graphical output features, e.g., for depicting three-dimensional functions, are quite advanced.

graphical frontend

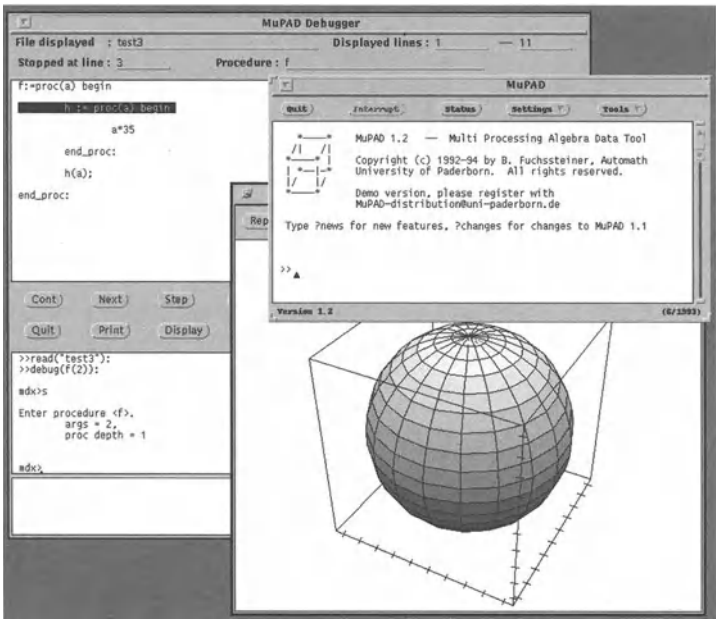


Fig. 11.18 Computer algebra system MuPAD

A hypertext on-line help function provides all relevant information and function descriptions. A detailed manual for MuPAD appeared with the publisher Birkhäuser in Basel (Switzerland) and Boston, MA.

Network Applications

The effective development and rapid dissemination of Linux is a phenomenon that would be inexplicable without the many communication features and services of the Internet. Since many of these features remain largely unknown outside university circles, we present a rough overview of the most important Internet services.

12.1 Electronic mail

The electronic mail system under UNIX usually consists of two components: a front end for reading and writing letters (e-mails) and a daemon (`smail` or `sendmail`) that runs in the background to handle delivery of the mails. A normal user has no contact with this daemon and simply uses a suitable front end.

2 components

One widespread program for reading and writing e-mails is `elm`, which is included in almost all Linux distributions and exists for most UNIX platforms. When `elm` is started for the first time, it creates two subdirectories in the user's home directory: a configuration directory and a directory to store e-mails that are received.

front end

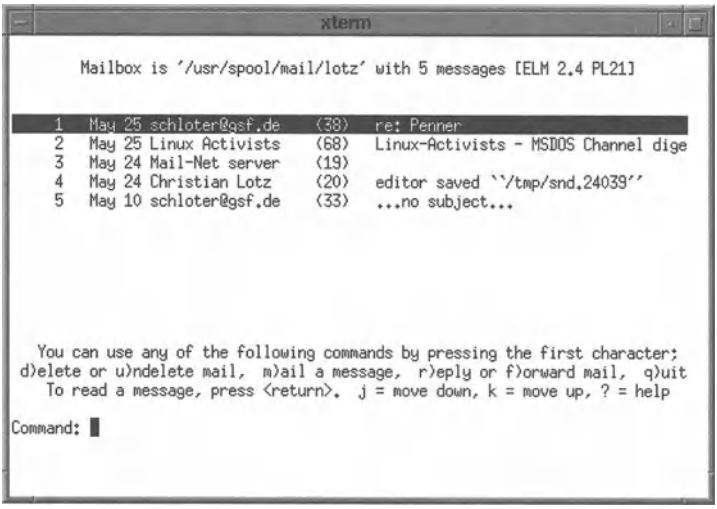


Fig. 12.1 Mail front end `elm`

New mails are displayed in a list by their sender and title. They can be selected with the cursor keys, read, and stored as files. For writing mails to be sent, the user defines an external editor in the `elm` options.

MIME

`elm` also handles mails in Multipurpose Internet Mail Extension (MIME) format, which can contain graphics, sounds, or programs. These MIME mails can be output with the `metamail` program and the respective presentation programs.

Pine, which stands for "Pine Is No longer Elm" or Program for Internet News & E-mail, is more comfortable than elm. A major difference with respect to elm is pine's feature allowing access to the Internet News (see Section 12.2).

IMAP

Pine contains an editor named pico that significantly eases the production of mails. Since pine uses the IMAP protocol, it can also manage mail folders that are located on a remote machine. This proves particularly interesting for a user who often switches among several machines. IMAP assures consistent access to the mails of the home machine and now supports the MIME standard as well.

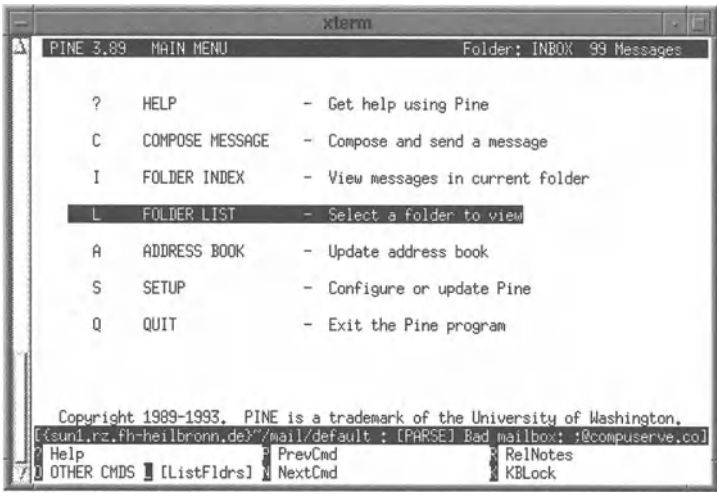


Fig. 12.2 Mail front end `pine`

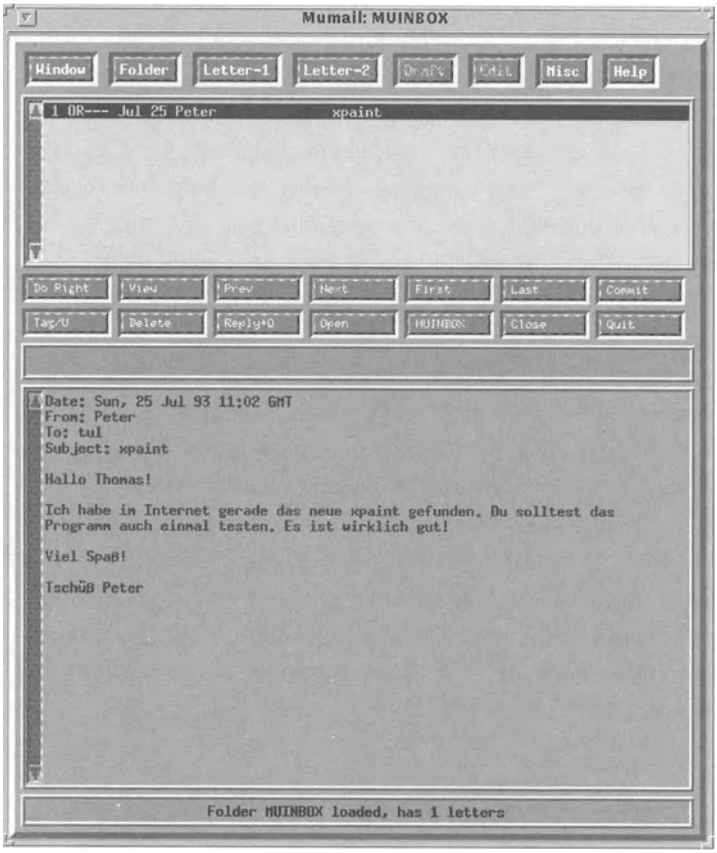


Fig. 12.3 Graphical mail front end MuMail

MuMail

An X11-based mail program with a graphical user interface and mouse operation, `MuMail` offers approximately the same functionality as `elm` and supports the creation and display of multimedia mails.

12.2 News

The Internet News is one of the most important sources of information about new developments and many other subjects.

Structure

The principle is quite simple. News servers have been set up at many locations around the Internet, for example, at almost every university. A news server manages various news groups, which amount to bidirectional electronic bulletin boards, each on a certain subject area, for posting, reading, and replying. These subjects range from operating systems and programs to sports, computer games, and other recreational activities, to matters of social interest. Several thousand such newsgroups exist worldwide (see below).

electronic bulletin boards

newsgroups

All news servers are constantly exchanging postings with neighboring news servers, so that a message posted on a given news server in a certain group quickly propagates to all news servers. These postings remain on the news servers for a certain time, for example, two weeks, before they are deleted.

The entirety of all computers that exchange these postings in the form of news is called USENET. Communication between these machines was originally based on modems and UUCP (UNIX to UNIX copy). This made it significantly slower than the current connections on the Internet, which use a special protocol called NNTP instead of UUCP.

USENET

UUCP

Newsgroups

The newsgroups are structured hierarchically; their full names, separated by periods, reflect their position in this hierarchy. `comp.os.linux.announce`, for example, means that this newsgroup resides under the heading `comp`, which deals with computers, software, and computer science in general. `os` indicates operating systems, and `comp.os.linux` is the leader for all names of groups under the `comp` hierarchy that have to do with Linux.

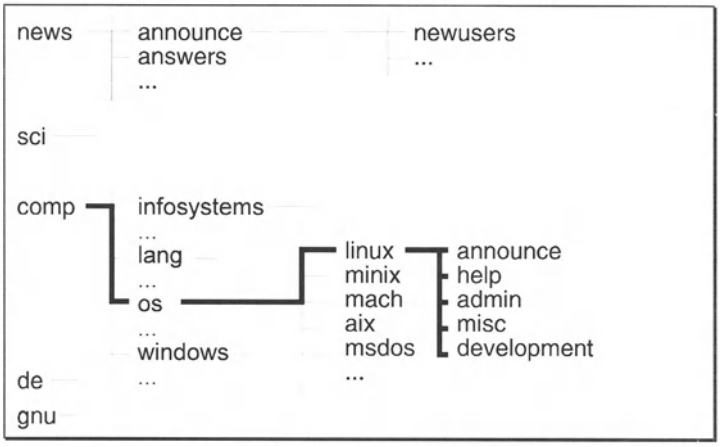


Fig. 12.4 A small excerpt from the newsgroup hierarchy

Besides `comp` there are many other important hierarchies such as `sci` for groups that discuss scientific subject matter and `alt` for a broad range of alternative subject areas.

Moderated groups

Groups can be either open to any USENET News participant or moderated. In the latter case a moderator decides which proposed

postings might be of interest to the readership of the newsgroup; these are posted, while the rest find their way into a black hole. For example, a moderated newsgroup for Linux is `comp.os.linux.announce`, which exclusively posts announcements such as new programs and system extensions. This is especially necessary because the number of postings in the other Linux groups is so overwhelming that a participant has to invest a great deal of time to read them and keep up to date.

Unmoderated groups are open, so that every participant can post statements, questions, and replies.

Rules and nettiquette

new groups

A new newsgroup is created if it is recommended by a USENET participant and the suggestion is accepted in a network vote. Discussions about proposed new groups often take place in the group `news.groups`.

When a participant wants to post a message to a newsgroup, certain rules should be observed, which constitute the network etiquette, or nettiquette. Insulting comments or personal attacks are not tolerated. The newsgroup `news.announce.newusers` contains a regular posting with an overview of these rules of network behavior.

News reader

Reading the USENET News requires a news reader and a news server that is accessible via network or modem. At universities this is usually no problem since they usually operate their own news server.

available options

News readers have become available for almost all computer platforms and operating systems that support a network. Well-known news readers include `rn`, `trn`, `tin`, `xrn`, and `xvnews` for the X Window System and `trumpet` for PCs. Almost all of these news readers exist under Linux.

We recommend using `xrn`, which offers a nice graphical user interface and the choice of the Athena widget version or the Motif variant.

Fig. 12.5 Newsreader `xrn`

12.3 Gopher

Gopher is an Internet service that presents itself to the user as a single, large, hierarchical menu. Within this structure the user can move from one server to another and employ various network services. The offer includes literature databases, weather services, all kinds of documents (text, pictures, sound), cafeteria menus at various universities, telnet sessions, and gateways to other services such as WAIS (see Section 12.6). An increasing number

hierarchical menu

of institutions are providing information via their Gopher service that previously was available only in printed form.

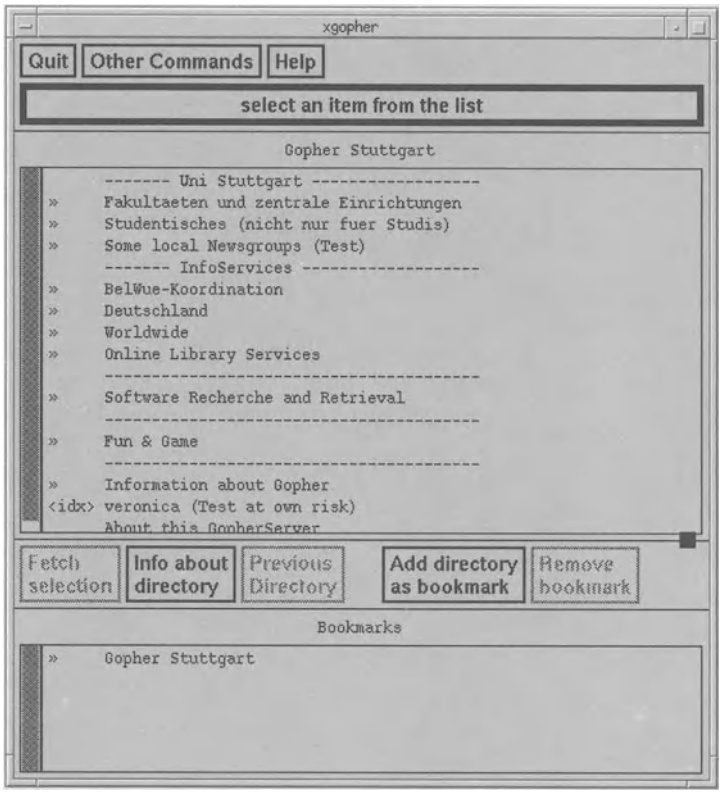


Fig. 12.6 Gopher client `xgopher`

XGopher

To be able to use Gopher, the user needs a Gopher client. Under Linux this could be the `xgopher` that runs under the X Window System. When it is started, `xgopher` establishes a connection to a Gopher server, and the user can move freely in the Internet.

12.4 IRC

Internet Relay Chat (IRC) offers a means to converse directly with several participants on the Internet. Similar to Internet News, IRC represents a system of networked servers that

exchange information among themselves. As with USENET News, there is a structure according to subjects, called *channels* under IRC.

real-time conversations

What distinguishes IRC from Internet News is that the exchange of messages occurs without delay and the user can converse directly with the other participants. This proves particularly interesting when there is an acute problem that needs to be discussed with others.

The user gains access with a special IRC client program that establishes the connection to the next IRC server. The IRC client has certain commands that all begin with a slash (`/`). These commands enable the user, for example, to sign on and off a channel. Text that is entered without a command is immediately transferred and is displayed for all other IRC participants that are currently signed on the same channel.

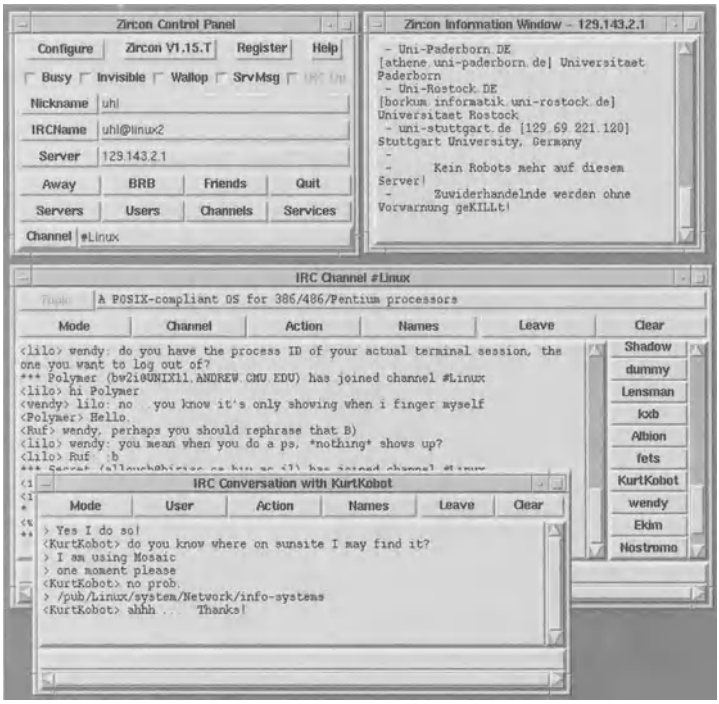


Fig. 12.7 Groupware tool `zircon` (IRC)

IRC also has a channel that is dedicated to Linux discussions. To take part in this discussion, start the IRC program and enter `/join #linux`. `/join` is the command to sign on a channel. Then every message that is written on the channel is displayed along with the name of the sender.

creating channels

The number of channels changes constantly, since every user can open new channels. This is also done with the `/join` command. If the specified channel does not exist, it is created. Normally hundreds of channel are active simultaneously; all currently active channels can be displayed with the `/list` command.

zircon

Besides its on-line discussion feature, IRC protocol also permits private messages and the exchange of smaller files. The graphical IRC client called zircon for the X Window System uses Tcl-dp (a Tcl interpreter that contains extensions for networking). It is available from `catless.ncl.ac.uk`; a compiled version is available from the usual Linux ftp servers.

12.5 Archie

Given the vast number of ftp servers and the huge amount of free software available on them, it can be quite difficult to find the right software to cover a particular requirement. This is where Archie servers provide valuable help.

software database

An Archie server furnishes access to a multiple-gigabyte database that contains an index of the most important ftp servers on the Internet. This database is updated automatically at regular intervals. Archie servers are located at the following addresses:

- `archie.ans.net` (ANS server, NY (USA))
- `archie.au` (Australian Server)
- `archie.doc.ic.ac.uk` (United Kingdom Server)
- `archie.edvz.uni-linz.ac.at` (Austrian Server)
- `archie.funet.fi` (Finnish Server)
- `archie.internic.net` (AT&T server, NY (USA))
- `archie.kr` (Korean Server)
- `archie.kuis.kyoto-u.ac.jp` (Japanese Server)
- `archie.luth.se` (Swedish Server)

- archie.ncu.edu.tw (Taiwanese server)
- archie.nz (New Zealand server)
- archie.rediris.es (Spanish Server)
- archie.rutgers.edu (Rutgers University (USA))
- archie.sogang.ac.kr (Korean Server)
- archie.sura.net (SURAnet server MD (USA))
- archie.sura.net (SURAnet alt. MD (USA))
- archie.switch.ch (Swiss Server)
- archie.th-darmstadt.de (German Server)
- archie.unipi.it (Italian Server)
- archie.univie.ac.at (Austrian Server)
- archie.unl.edu (U. of Nebraska, Lincoln (USA)
- archie.uqam.ca (Canadian Server)
- archie.wide.ad.jp (Japanese Server)

To submit a query to an Archie server, the user logs in to one of the respective hosts with telnet with the user name archie and uses a simple query language to search for programs.

xarchie

The graphical front end xarchie, also available under Linux, provides this service with more comfort. The user only needs to enter the keyword and select the search mode, and xarchie establishes the connection to the set-up server. Soon the files and their server locations are displayed in a browser. Newer versions of xarchie can even start an ftp transfer directly to retrieve the desired files from their respective ftp servers.

whatis

In addition to searching for programs by name, an Archie server provides a whatis database in which the whatis command yields a concise description of a specific program. If the name of a program is not known at all or only in part, whatis can display a list of a relevant subset of all registered programs by name along with a description of each.

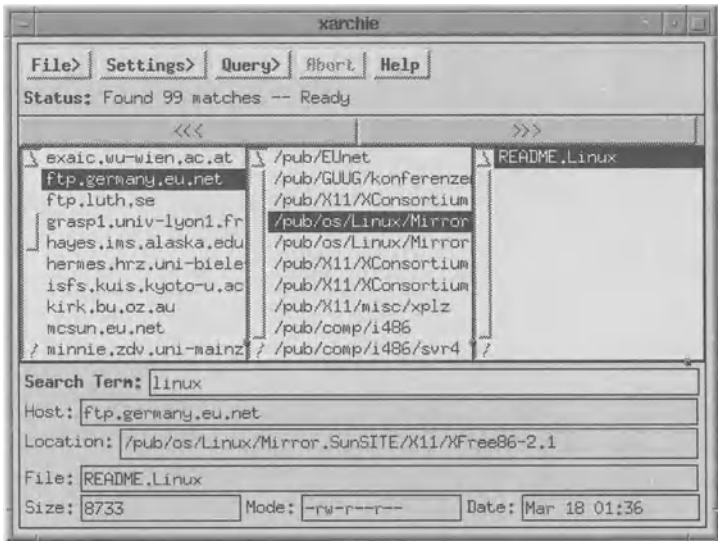


Fig. 12.8 Archie frontend `xarchie`

e-mail

In lieu of Internet access, an Archie server can be queried per e-mail. Here the appropriate command is sent to the user `archie` on such a server. As with ftp mail servers, sending the command `help` to the Archie server in the first line of the mail returns a more detailed description.

12.6 WWW and Mosaic

Mosaic is a very comfortable program that supports access to many of the above Internet services from a common user interface. Supported services include Gopher, News, World Wide Web (WWW), Wide Area Information Service (WAIS), Archie, and ftp.

WWW

WWW is an Internet service that employs distributed multimedia hypertext documents that can contain not only formatted texts but also references to other documents, pictures, videos, or sound files. These documents are described in the

HTML, HTTP

Hypertext Markup Language (HTML) in ASCII format. A special protocol called Hypertext Transfer Protocol (HTTP) is used for data transfer.

References in HTML documents are specified as Uniform Resource Locators (URLs), which use the form "`protocol://host address:port/path`". Specification of the port number is optional; if no port is specified, the standard port for the specified protocol is used.

Such URLs permit the expression of not only references to other HTML files but also almost all kinds of addresses of services on the Internet. For example, the URL for the directory `/pub/Linux` on the ftp server `sunsite.unc.edu` is: `ftp://sunsite.unc.edu /pub/Linux`.

Compiling this program requires OSF/Motif, which is not freeware. However, there is also a compiled version that is statically linked and thus does not need the OSF/Motif run-time system.

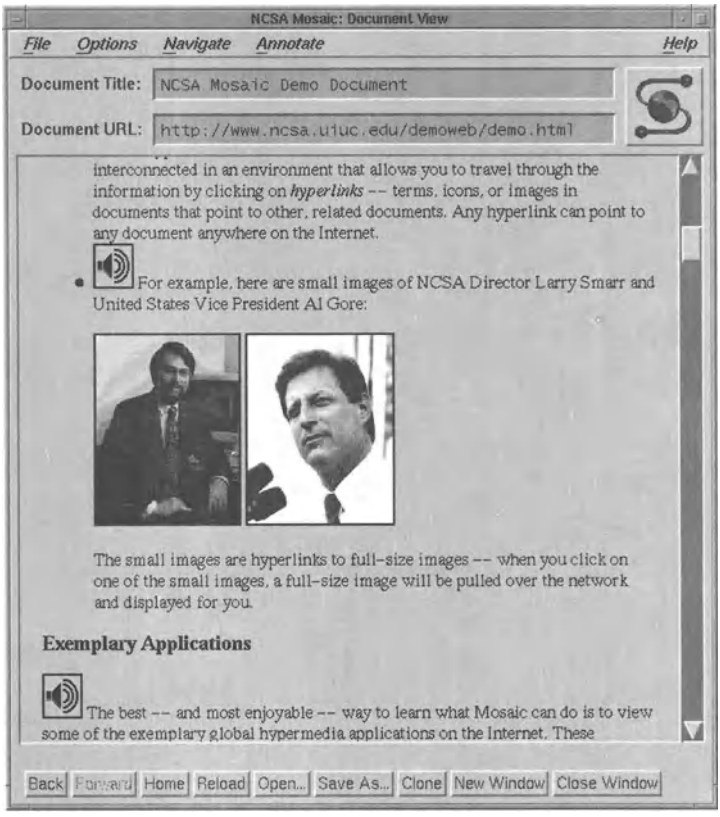


Fig. 12.9 WWW access via Mosaic

With the help of an HTTP server, a distributed information system can be constructed easily under Linux. Such a server, running in the background as a daemon, waits for a socket connection via a defined port; after a connection has been established by a WWW client (e.g., Mosaic), the server furnishes access to the files in certain directories. If direct Internet access is available, the local documents can offer appropriate links to other HTTP servers.

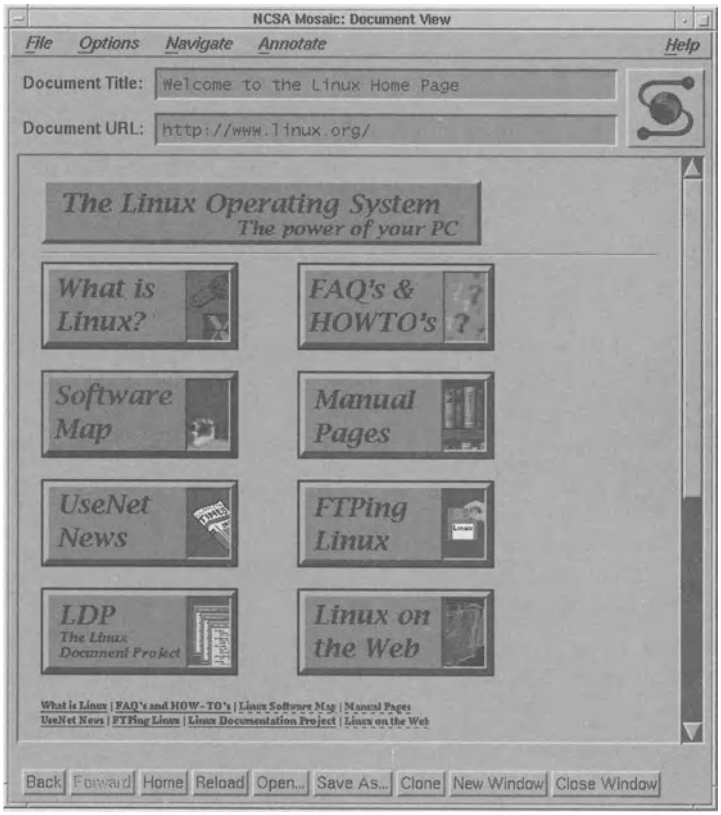


Fig. 12.10 HTTP server with current information on Linux

Interesting servers for Linux users can be found at the following URLs:

- http://linux.org/
- http://sunsite.unc.edu/mdw/linux.html
- http://www.fokus.gmd.de/linux/
- http://wwwi2.informatik.rwthaachen.de/Linux.html
- http://www.informatik.uni-dortmund.de/IRB/Linux/
- http://www.fintronic.com/linux/catalog.html

Appendix

This appendix provides an overview of the configuration files and subdirectories located in the important directory `/etc`. In addition, a sample configuration of the kernel furnishes a reference for Linux users.

Overview of `/etc` files

Most configuration files reside in the directory `/etc`. This section gives an overview of the most important of these configuration files. Our list refers primarily to the Slackware distribution, which does not completely adhere to the new Linux File System Standard (FSSTND).

new file system standard

- **`bootptab`** - configuration file for bootpd daemon
- **`csh.cshrc`** - global definition of (t)chs
- **`chs.login`** - global login shell definitions for (t)chs
- **`diphosts`** - list of machines permitted to make a SLIP connection
- **`disktab`** - file for determination of abnormal hard disk parameters for LILO
- **`DIR_COLORS`** - configuration file with color settings of the ls command
- **`exports`** - This file defines the directories to be exported by NFS.
- **`fdprm`** - parameters for floppy disk drives
- **`fstab`** - This file contains the file systems and swap partitions to be mounted or activated on booting .

floppy disk

- **ftpaccess** - This configuration file for the ftp daemon allows the definition of access restrictions and messages.
- **ftpusers** - This is the configuration file for the ftp daemon. Users listed in this file cannot log in per ftp.

login
- **gateways** - file with a list of gateways
- **gettydefs** - configuration file for `getty`

groups
- **group** - This file lists all groups and their member users. If a user belongs to multiple groups, then the user name appears with multiple groups. The only group name contained in the `passwd` file is the user's primary group name.
- **gshadow** - shadow file for the file `group`

address resolution
- **host.conf** - This file specifies the sequence in which to search for an IP address. `order hosts bind`, for example, means that first the file hosts will be searched, and if the desired host is not entered there, the search continues on the name server.
- **hosts** - This file contains the names and IP addresses of other computers. Depending on the settings in the file `host.conf`, this file is searched before or after the name server to find the IP address of a host.
- **hosts.allow** - This file lists computers that are allowed to access local network services (tcp wrapper).
- **hosts.deny** - This file lists computers that are denied access local network services (tcp wrapper).
- **hosts.equiv** - This file defines trusted users/hosts that for the Berkeley r-utilities.
- **hosts.lpd** - This file lists computers that are allowed to use local printers.

TCP/IP
- **inetd.conf** - This is a configuration file for the daemon `inetd`. It establishes the one-to-one correspondence between a connection to a certain port and the daemon to be started.
- **inittab** - This configuration file for the `init` process contains the assignments of run levels.

login prompt
- **issue** - This is the text file which is displayed before the login prompt. Normally this file contains the name of the host and a greeting message.

- **login.defs** - This configuration file for `login` defines the devices from which it is possible to log in as `root` and which text files are to be displayed after a login. Like the shadow files, this file is used by the shadow file suite. This file does not exist in the Slackware distribution, which does not use shadow passwords.
- **ld.so.conf** - file with paths to shared libraries
- **lilo.conf** - configuration file for Linux Loader (LILO)
- **magic** - This file establishes the correspondence between byte patterns at the beginning of a file and the respective file types. A file's type can be determined by searching in this file for its starting byte pattern. — file type
- **motd** - This text file (message of the day) is normally displayed automatically with each login. — login message
- **mtab** - This internal table for `mount` lists all currently mounted file systems. — file systems
- **mtools** - configuration file for mtools
- **named.boot** - boot file for the name server
- **networks** - This file contains the IP addresses of known networks.
- **passwd** - Users are defined in this file by specifying their user IDs, user names, login shells, and primary groups. The actual passwords are stored in the file `shadow`, which, unlike the `passwd` file, can only be read with root permissions. However, if the shadow password package is not installed, then the passwords are stored here. — users and passwords
- **printcap** - configuration file for printer queues — printer queue
- **profile** - This global profile file for Bourne/Korn/bash shells is executed automatically on startup. Paths and variables for all users are defined here.
- **protocols** - This file defines the TCP/IP packet types.
- **resolv.conf** - This configuration file for the TCP/IP system contains the local domain name and the address of the name server.
- **rpc** - configuration file for RPC programs
- **securetty** - This file specifies on which TTYs the superuser can log in.

- **services** - This file contains the assignment of TCP/IP ports to corresponding names of services, which can be assigned to daemons in the file `inetd.conf` or can be used by other programs.
- **shadow** - file with encrypted user passwords
- **shells** - This file contains the valid system shells. When the user logs in with ftp, the ftp daemon checks whether the user's login shell is entered in this list. If not, the login is denied.
- **syslog.conf** - configuration file for the syslog daemon
- **termcap** - This configuration file for the `termcap` library defines the functions of the various types of terminals.
- **utmp, wtmp** - log file of all user logins and logouts
- **xmmounttab** - configuration file for `xmmount`
- **xvmounttab** - configuration file for `xvmount`

Overview of `/etc` subdirectories

- **/etc/default** - This directory stores the default values of parameters
- **/etc/default/useradd** - file with default values for useradd command
- **/etc/default/getty.XXX** - configuration of `getty` program for port XXX
- **/etc/default/uugetty** - configuration of `uugetty` program for port XXX
- **/etc/skel** - This directory contains files that, on execution of `useradd`, are automatically copied into the home directory of a new user.
- **/etc/rc.d** - This directory contains scripts that are invoked by init when the run level of the entire system changes.
- **/etc/rc.d/rc.0** - script for run level 0 (system shutdown)
- **/etc/rc.d/rc.6** - script for run level 6 (login via `xdm`)
- **/etc/rc.d/rc.K** - script for single user mode
- **/etc/rc.d/rc.M** - script for multi-user mode (run levels 1 to 6)

- `/etc/rc.d/rc.S` - script that activates swapping and checks consistency of file system on booting
- `/etc/rc.d/rc.keymap` - script for loading country-specific keyboard layouts
- `/etc/rc.d/rc.local` - script to start local daemons
- `/etc/rc.d/rc.inet1` - script to initialize network interfaces
- `/etc/rc.d/rc.inet2` - script to start network daemons
- `/etc/rc.d/rc.serial` - script to configure serial interfaces

Configuration of the kernel

The following listing shows an exemplary configuration of the Linux kernel. Since the kernel remains in a state of flux, deviations are possible.

```
bigmac:/usr/src/linux# make config
/bin/sh Configure  < config.in
*
* General setup
*
Kernel math emulation (CONFIG_MATH_EMULATION) [n]
Normal harddisk support (CONFIG_BLK_DEV_HD) [y]
XT harddisk support (CONFIG_BLK_DEV_XD) [n]
TCP/IP networking (CONFIG_INET) [y]
Limit memory to low 16MB (CONFIG_MAX_16M) [n]
System V IPC (CONFIG_SYSVIPC) [y]
Use -m486 flag for 486-specific optimizations (CONFIG_M486) [y]
*
* Program binary formats
*
Elf executables (CONFIG_BINFMT_ELF) [n]
COFF executables (CONFIG_BINFMT_COFF) [n]
*
* SCSI support
*
SCSI support? (CONFIG_SCSI) [y]
        *
        * SCSI support type (disk, tape, CD-ROM)
        *
SCSI disk support (CONFIG_BLK_DEV_SD) [y]
SCSI tape support (CONFIG_CHR_DEV_ST) [y]
SCSI CDROM support (CONFIG_BLK_DEV_SR) [y]
SCSI generic support (CONFIG_CHR_DEV_SG) [n]
        *
        * SCSI low-level drivers
        *
Adaptec AHA152X support (CONFIG_SCSI_AHA152X) [n]
Adaptec AHA1542 support (CONFIG_SCSI_AHA1542) [y]
Adaptec AHA1740 support (CONFIG_SCSI_AHA1740) [n]
Future Domain 16xx SCSI support (CONFIG_SCSI_FUTURE_DOMAIN) [n]
Generic NCR5380 SCSI support (CONFIG_SCSI_GENERIC_NCR5380) [n]
PAS16 SCSI support (CONFIG_SCSI_PAS16) [n]
Seagate ST-02 and Future Domain TMC-8xx SCSI support
(CONFIG_SCSI_SEAGATE) [n]
Trantor T128/T128F/T228 SCSI support (CONFIG_SCSI_T128) [n]
UltraStor SCSI support (CONFIG_SCSI_ULTRASTOR) [n]
7000FASST SCSI support (CONFIG_SCSI_7000FASST) [n]
*
* Network device support
*
Network device support? (CONFIG_ETHERCARDS) [y]
SLIP (serial line) support (CONFIG_SLIP) [y]
 CSLIP compressed headers (SL_COMPRESSED) [y]
PLIP (parallel port) support (CONFIG_PLIP) [y]
NE2000/NE1000 support (CONFIG_NE2000) [n]
WD80*3 support (CONFIG_WD80x3) [n]
SMC Ultra support (CONFIG_ULTRA) [y]
3c501 support (CONFIG_EL1) [n]
3c503 support (CONFIG_EL2) [n]
3c509/3c579 support (CONFIG_EL3) [n]
HP PCLAN support (CONFIG_HPLAN) [n]
AT1500 and NE2100 (LANCE and PCnet-ISA) support (CONFIG_LANCE)
[n]
AT1700 support (CONFIG_AT1700) [n]
DEPCA support (CONFIG_DEPCA) [n]
D-Link DE600 pocket adapter support (CONFIG_DE600) [n]
AT-LAN-TEC/RealTek pocket adapter support (CONFIG_ATP) [n]
*
Sony CDU31A CDROM driver support (CONFIG_CDU31A) [n]
Mitsumi CDROM driver support (CONFIG_MCD) [n]
Matsushita/Panasonic CDROM driver support (CONFIG_SBPCD) [n]
*
* File systems
*
Standard (minix) fs support (CONFIG_MINIX_FS) [y]
Extended fs support (CONFIG_EXT_FS) [n]
Second extended fs support (CONFIG_EXT2_FS) [y]
xiafs file system support (CONFIG_XIA_FS) [n]
msdos fs support (CONFIG_MSDOS_FS) [y]
/proc file system support (CONFIG_PROC_FS) [y]
```

```
NFS file system support (CONFIG_NFS_FS) [y]
ISO9660 CD-ROM file system support (CONFIG_ISO9660_FS) [y]
OS/2 HPFS file system support (read only) (CONFIG_HPFS_FS) [n]
System V and Coherent file system support (CONFIG_SYSV_FS) [n]
*
*  character devices
*
Parallel printer support (CONFIG_PRINTER) [y]
Logitech busmouse support (CONFIG_BUSMOUSE) [n]
PS/2 mouse (aka "auxiliary device") support (CONFIG_PSMOUSE)
[n]
Microsoft busmouse support (CONFIG_MS_BUSMOUSE) [n]
ATIXL busmouse support (CONFIG_ATIXL_BUSMOUSE) [n]
Selection (cut and paste for virtual consoles)
(CONFIG_SELECTION) [n]
QIC-02 tape support (CONFIG_TAPE_QIC02) [n]
QIC-117 tape support (CONFIG_FTAPE) [n]
*
* Sound
*
Sound card support (CONFIG_SOUND) [y]
*
* Kernel hacking
*
Kernel profiling support (CONFIG_PROFILE) [n]
Verbose scsi error reporting (kernel size +=12K)
(CONFIG_SCSI_CONSTANTS) [y]

The linux kernel is now hopefully configured for your setup.
Check the top-level Makefile for additional configuration,
and do a 'make dep ; make clean' if you want to be sure all
the files are correctly re-made

CONFIG_SOUND = CONFIG_SOUND
make[1]: Entering directory `/usr/src/linux/drivers/sound'
Compiling Sound Driver v 2.4 for Linux

Configuring the sound support

Do you want to include full version of the sound driver (n/y) ?
n
Do you want to DISABLE the Sound Driver (n/y) ?n
The SoundBlaster, AdLib and ProAudioSpectrum
cards cannot be installed at the same time

Select at most one of them:
        - ProAudioSpectrum 16
        - SoundBlaster / SB Pro
          (Could be selected with PAS16 also
          since there is a SB emulation on it)
        - AdLib

Don't enable SoundBlaster if you have GUS at 0x220!

  ProAudioSpectrum 16 support (n/y) ? n
  SoundBlaster support (n/y) ? y

The following cards should work with any other cards.
CAUTION! Don't enable MPU-401 if you don't have it.
  Gravis Ultrasound support (n/y) ? n
  MPU-401 support (NOT for SB16) (n/y) ? n

Select one or more of the following options
  SoundBlaster Pro support (y/n) ? y
  SoundBlaster 16 support (y/n) ? y
  digitized voice support (y/n) ? y
  MIDI interface support (y/n) ? y
  FM synthesizer (YM3812/OPL-3) support (y/n) ? y

IRQ number for SoundBlaster?
The IRQ address is defined by the jumpers on your card and
7 is the factory default. Valid values are 9, 5, 7 and 10.
Enter the value:
SoundBlaster IRQ set to 7

DMA channel for SoundBlaster?
```

```
For SB 1.0, 1.5 and 2.0 this MUST be 1
SB Pro supports DMA channels 0, 1 and 3 (jumper)
For SB16 give the 8 bit DMA# here
The default value is 1
Enter the value:
SoundBlaster DMA set to 1

16 bit DMA channel for SoundBlaster 16?
Possible values are 5, 6 or 7
The default value is 6
Enter the value:
SoundBlaster DMA set to 6

I/O base for SB16 Midi?
Possible values are 300 and 330
The factory default is 330
Enter the SB16 Midi I/O base:
SB16 Midi I/O base set to 330

Select the DMA buffer size (4096, 16384, 32768 or 65536 bytes)
65536 is recommended value for this configuration.
Enter the value:
The DMA buffer size set to 65536
The sound driver is now configured.
make[1]: Leaving directory `/usr/src/linux/drivers/sound'
mv .tmpconfig .config
bigmac:/usr/src/linux#
```

Further reading

Andeleigh, Prabhat K.
UNIX System Architecture
Prentice Hall [1993]

Comer, Douglas E.
Internetworking with TCP/IP, Vol. 1-3
Prentice Hall [1991]

Faith, Rik (faith@cs.unc.edu)
Linux Man Pages/Online Documentation
Linux Documentation Project

Flanagan, David
X Toolkit Intrinsics Reference Manual, Vol. 5
O'Reilly [1993]

Fuchssteiner, Beno
MuPAD - User's maual
Birkhäuser [1994]

Gilly, Daniel
UNIX in a Nutshell
O'Reilly [1992]

Greenfield, Larry (greenfie@gauss.rutgers.edu)
Linux User's Guide
Linux Documentation Project

Heller, Dan
XView Programming Manual, Vol. 7
O'Reilly [1991]

Heller, Dan
Motif Programming Manual, Vol. 6 A/B
O'Reilly [1994]

Hewlett Packard
Ultimate Guide to the Vi and Ex Text Editors
Addison Wesley [1990]

Johnson, Michael K. (johnsonm@sunsite.unc.edu)
Linux INFO-SHEET
Linux Documentation Project

Johnson, Michael K. (johnsonm@sunsite.unc.edu)
Linux Kernel Hacker's Guide
Linux Documentation Project

Johnson, Michael K. (johnsonm@sunsite.unc.edu)
Linux META-FAQ
Linux Documentation Project

Kernighan, B. and Pike, R.
The UNIX Programming Environment
Prentice Hall [1984]

Kirch, Olaf (okir@mathematik.th-darmstadt.de)
Linux Network Administrator's Guide
Linux Documentation Project

Kopka, Helmut and Daly, Patrick W.
A Guide to Latex: Document Preparation for Beginners & Advanced Users
Addison Wesley [1993]

Lippmann, Stanley B.
C++ Primer
Addison Wesley [1991]

Mui, Linda und Pearce, Eric
X Window System Administrator's Guide, Vol. 8
O'Reilly [1993]

Nye, Adrian
Xlib Programming Manual, Vol. 1
O'Reilly [1992]

Nye, Adrian
Xlib Reference Manual, Vol. 2
O'Reilly [1992]

Nye, Adrian and O'Reilly, Tim
X Toolkit Intrinsics Programming Manual, Vol. 4
O'Reilly [1990]

Open Software Foundation
OSF/Motif Progammer's Guide
Prentice Hall [1993]

Open Software Foundation
OSF/Motif Progammer's Reference
Prentice Hall [1993]

Open Software Foundation
OSF/Motif Users's Guide
Prentice Hall [1993]

Quercia, Valerie and O'Reilly, Tim
X Window System User's Guide, Vol. 3
O'Reilly [1991]

Santifaller, Michael and Wilson, Stephen S. (Translator)
Tcp/IP and NFS: Internetworking in a Unix Environment
Addision Wesley [1991]

Schoonover, Michael A.
GNU Emacs - UNIX Text Editing and Programming
Addison Wesley [1992]

Sobell, Mark
A Practical Guide to UNIX System V, 2nd edition
Benjamin-Cummings [1991]

Stevens, W. Richard
Advanced Programming in the UNIX Environment
Addison Wesley [1992]

Stevens, W. Richard
UNIX Network Programming
Prentice Hall [1990]

Welsh, Matt (mdw@sunsite.unc.edu)
Linux HOWTO Documents
Linux Documentation Project

Welsh, Matt (mdw@sunsite.unc.edu)
Linux Installation and Getting Started
Linux Documentation Project

Wirzenius, Lars (wirzeniu@cc.helsinki.fi)
Linux System Administrator's Guide
Linux Documentation Project

Young, Douglas A.
X Window System, Programming and Applications with X
Prentice Hall [1990]

Index